Universidad Carlos III de Madrid
www.uc3m.es

Desarrollo de un prototipo de realidad virtual para pacientes con trastornos neurológicos

---

TRABAJO FIN DE GRADO

Autor: Jorge García Puig

Titulación: Grado en Ingeniería Informática

Tutor: Yago Sáez Achaerandio

Fecha de entrega: 06-07-2020

Fecha de presentación: 23-07-2020

ESCUELA POLITÉCNICA SUPERIOR

AGRADECIMIENTOS

Para todas las personas que me apoyaron en este proyecto, en especial a mi familia, por el apoyo emocional que me han proporcionado durante todo el proyecto. A mi tutor D. Yago Sáez Achaerandio por haberme dado la posibilidad de emprender este proyecto. Y a Alejandro Baldominos y Jesús Mayor por la simpatía que ofrecieron durante la cita presencial para prestarme los visores.

RESUMEN

Los trastornos neurológicos son enfermedades que afectan al sistema nervioso y, con ello, al cerebro. Generalmente, los trastornos alteran en la movilidad y percepción de los sentidos provocando pérdida del equilibrio, parálisis, incapacidad del habla, etc. El videojuego desarrollado se utilizará como un ejercicio de rehabilitación para entrenar la capacidad reducida del paciente a causa de una enfermedad neurológica.

El videojuego funcionará en realidad virtual y consistirá en la destrucción de una secuencia de obstáculos que se irán mostrando al paciente. El paciente deberá tocar cada obstáculo de la manera más rápida y precisa posible. Con los resultados de la prueba, se podrá obtener información de interés del paciente: reflejos, capacidad motora, agilidad, capacidad reactiva, procesamiento de imágenes y dificultad para orientarse.

El videojuego cumple con una serie de restricciones y requisitos iniciales y adicionales para cumplir con los objetivos deseados. A partir de un estudio de las diferentes alternativas de diseño que cumplan con los requisitos establecidos, se seleccionará un modelo de diseño.

Para el modelo de diseño escogido, se explicará progresivamente el procedimiento llevado a cabo hasta su funcionamiento final. En primer lugar, se describirá el modelado del entorno y de todos los objetos tridimensionales creados y, finalmente la implementación.

Debido a la cantidad de recursos necesarios para la simulación de entornos de realidad virtual, el videojuego resultante se ha dividido en una versión optimizada y otra no optimizada. Así, se puede garantizar un funcionamiento fluido del videojuego tanto para jugadores con equipos de alta gama como para jugadores con equipos de media gama.

La versión optimizada se ha utilizado para probar el videojuego con personas reales y así recoger una muestra inicial de resultados acerca del funcionamiento del sistema. A través de estos resultados, se ha observado pacientes con dificultades motoras en un brazo concreto y su disminución de la capacidad reactiva y de reflejos.

El desarrollo del proyecto se ha realizado siguiendo una planificación establecida inicialmente para cada tarea. Esto ha permitido la evaluación independiente de cada tarea a través de un método de evaluación. De esta forma, se consigue mayor precisión en la puntuación media adquirida por el cumplimiento de los objetivos.

Finalmente, el prototipo resultante se encuentra en una versión estable y funcional. Esta versión se podría utilizar en centros de salud u hospitales para medir capacidades de los pacientes.

PALABRAS CLAVE

Realidad virtual, trastorno neurológico, Unity, videojuego, diseño.

ABSTRACT

Neurological disorders are diseases that affect the nervous system and the brain. Generally, these disorders alter the mobility and perception of the senses causing loss of balance, paralysis, speech disability, etc... The developed videogame will be used as a rehabilitation exercise to train the patient's reduced capacity due to neurological disease.

The videogame will work in virtual reality and will consist in the destruction of a sequence of obstacles that will be displayed to the patient. The patient must touch each obstacle as fast and as accurate as possible. With the results of the test, it will be possible to obtain information of interest to the patient: reflexes, motor ability, agility, reactive capacity, image processing and difficulty in orientation.

The videogame complies a set of initial and additional restrictions and requirements to satisfy the desired objectives. A design model will be selected, from a study of the different design alternatives that comply the established requirements.

For the chosen design model, the procedure will be progressively explained. First, the modeling of the environment and all the three-dimensional objects created will be explained, and finally the implementation.

The resulting videogame has been divided into an optimized version and a non-optimized version due to the amount of resources that virtual reality requires. Thus, a fluid videogame can be guaranteed for both players with powerful computer and medium-range computers.

The optimized version has been used to test the videogame with real people and focus on analyzing its results. Through these results, patients with motor difficulties in a specific arm and decreased reaction and reflex capacity have been considered.

The development of the project has been carried out following a planning established initially for each task. This has allowed the independent evaluation of each task through an evaluation method. In this way, greater precision is achieved in the average score acquired for satisfying the objectives.

Finally, the resulting prototype is a stable and functional version. This version could be used in health centers or hospitals to measure patient capacities.

KEYWORDS

Virtual Reality, neurological disorder, Unity, videogame, design.

Índice de contenido

Índice de figuras

Índice de tablas

Índice de gráficas

1. INTRODUCCIÓN Y OBJETIVOS

En este trabajo se pretende crear un prototipo de videojuego de realidad virtual para entrenar la coordinación motora en pacientes con problemas tras un trauma neurológico.

1.1. Introducción

Los trastornos neurológicos son enfermedades que afectan al sistema nervioso y, con ello, al cerebro. Los síntomas de un trastorno neurológico son muy variados dependiendo del tipo de persona y de la enfermedad. Generalmente, los trastornos alteran en la movilidad y percepción de los sentidos provocando pérdida del equilibrio, parálisis, incapacidad del habla, etc. La duración de estos períodos es muy variada dependiendo de la fase en la que se encuentre.

Para la recuperación tras un trastorno neurológico es muy importante el diagnóstico y el tratamiento del paciente con la mayor antelación posible. Este trabajo se centrará en la creación de un ejercicio de rehabilitación para facilitar ese proceso de recuperación del paciente.

1.2. Objetivos y motivación

La motivación por realizar este proyecto viene principalmente por las personas que están sufriendo traumas neurológicos. Seguido a esto, la realidad virtual ha mejorado considerablemente desde sus orígenes y actualmente consigue obtener simulaciones lo suficientemente realistas como para abarcar diferentes áreas: educación, psicoterapia, conducción…

Los objetivos principales de este trabajo son:

1. Desarrollar un prototipo completamente funcional de un videojuego de realidad virtual desde su concepción hasta sus pruebas finales.
2. Aprender a utilizar el entorno Unity para el desarrollo de los requisitos funcionales.
3. Aprender a utilizar el programa Autodesk 3DS MAX para el desarrollo del modelado de los objetos tridimensionales utilizados.

Los objetivos más técnicos que también debe cumplir el videojuego del alumno son:

- El videojuego debe permitir su ejecución fluida para ordenadores más concretamente debe compatible con los sistemas operativos Windows y Macintosh puesto que son los más utilizados.
- La fluidez de un videojuego influye en el grado de conformidad del jugador, por tanto, el videojuego debe ser lo más fluido posible.
- El videojuego no deberá presentar fallos que impidan el progreso del mismo.
- El arte utilizado para los entornos debe ser agradable, para ello, gráficamente se debe entender lo que representa cada elemento de la escena.
- El videojuego debe tener una interfaz de menú principal antes de realizar la prueba.

- La interfaz presentada debe ser intuitiva para el usuario final. No debe influir el nivel de experiencia que tenga el usuario en los videojuegos.
- El videojuego debe recoger los datos de la interacción con el usuario para su posterior análisis por el personal sanitario.

1.3. Estructura de la memoria

La memoria se muestra de manera estructurada explicando progresivamente el desarrollo del proyecto. El capítulo 1 trata sobre los objetivos y la motivación por haber realizado este proyecto. El capítulo 2 presenta el planteamiento del problema acompañado del análisis del estado del arte donde se explica la evolución histórica de la computación gráfica y la realidad virtual, los motores gráficos más utilizados actualmente junto con su comparativa, la elección de uno de ellos y una línea del tiempo de la industria de los videojuegos. También se explicarán los requisitos que debe cumplir el videojuego planteado. En el capítulo 3 se explica el diseño de la solución técnica del videojuego, primero comienza con la idea inicial del videojuego, los elementos que puede contener un nivel y el software, técnicas y métodos planteados. Entonces, se barajarán diferentes alternativas de diseño y tras una comparativa entre ellas se seleccionará un modelo de diseño. Finalmente, se lleva a cabo la creación del modelo de diseño seleccionado pasando por diferentes fases: explicación del funcionamiento principal, diseño, modelado y animación de los objetos tridimensionales, implementación del funcionamiento, creación del menú principal, interfaz de usuario, inserción del sonido y optimización.

Una vez desarrollado el videojuego, se da comienzo con el capítulo 4. En este capítulo se lleva a cabo la explicación del método de evaluación del videojuego para conseguir puntuar y analizar el cumplimiento de las tareas y objetivos deseados. De esta forma, se obtendrá una puntuación media en la evaluación de los objetivos. En el capítulo 5 se dará comienzo con las pruebas del videojuego con jugadores reales, para ello, se explicará el protocolo realizado para las pruebas, los jugadores con los que se ha probado y un análisis de sus resultados conforme a las condiciones de las pruebas. En el capítulo 6 se mostrará la planificación realizada al inicio del proyecto y la resultante revisada al final del proyecto junto con su respectivo cronograma de actividades y su diagrama de Gantt. En el capítulo 7 se mostrarán los aspecto económicos y legales, así como un resumen de la repercusión económica generada por los videojuegos junto al presupuesto del proyecto y sus restricciones legales. Finalmente, el capítulo 8 cerrará con las conclusiones finales alcanzadas y los problemas encontrados.

2. ANÁLISIS DEL PROBLEMA

2.1. Introducción

El propósito de esta sección es realizar un análisis completo de las soluciones técnicas disponibles y de los requisitos del sistema que permitan justificar el motor gráfico y la plataforma del videojuego seleccionados. Primero se mostrará el progreso histórico que se ha seguido en la computación gráfica y en la realidad virtual, posteriormente se presentará una comparativa de los motores gráficos más conocidos actualmente y una línea del tiempo de la industria de los videojuegos. Finalmente se expondrán los requisitos del problema y las conclusiones alcanzadas.

2.2. Análisis del estado del arte

2.2.1. Introducción

El estado del arte en los videojuegos y en la realidad virtual no siguió un crecimiento instantáneo. Primero tuvieron que aparecer las primeras interfaces para facilitar la interacción del usuario con el ordenador y, posteriormente con las mejoras de los recursos hardware se pudieron fabricar computadores con mejores recursos con programas software gráficos como los de hoy día.

2.2.2. Evolución de la computación gráfica

La computación gráfica es el área de la informática visual que se centra en la proyección a través de una pantalla de imágenes generadas por un ordenador.

En el año 1946, en el Laboratorio de Investigación Balística del Ejército de los Estados Unidos se estaban realizando estudios con una bomba de nitrógeno. Debido a la peligrosidad de los estudios y la necesidad de un cálculo preciso de densidad, se creó la computadora ENIAC. Esta computadora mostraba los resultados mediante impresos en papel y en algunos casos con secuencias de código alfanumérico.

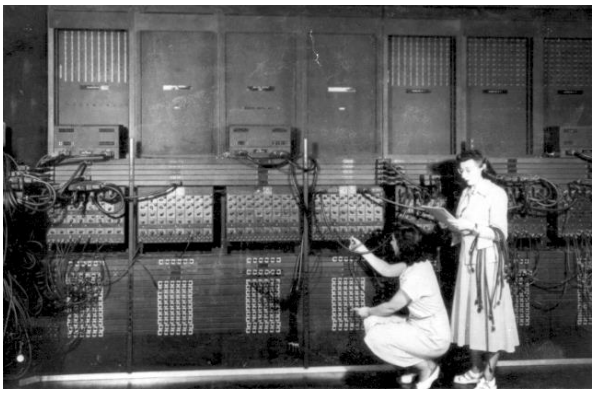


Fig.1 – ENIAC en Estados Unidos [1].

A raíz de aumentar la cantidad de los impresos de resultados y facilitar la traducción de las secuencias, comenzó a popularizarse la necesidad de cambiar la exposición de los resultados a monitores de computador donde el usuario pudiese interactuar con el programa de forma visual e intuitiva [1].

Años antes, Vannevar Bush se había dado cuenta de que el número de publicaciones de investigaciones había crecido y el ser humano no tenía capacidad suficiente para almacenar toda esa información. Además, los sistemas que se utilizaban para compartir la información eran demasiado lentos. Por tanto, el problema principal que Bush encuentra es que, cuando un investigador publica una tesis o una investigación que ha realizado, es muy probable que otros investigadores publiquen posteriormente otra investigación similar a la primera porque todavía no había llegado los resultados al público. Para paliar este problema, Bush concluye en la necesidad de

creación de MEMEX, un artefacto capaz de almacenar teóricamente enormes cantidades de documentos para lo que era aquella época.

MEMEX constaba de una mesa con palancas y motores que permitirían encontrar los documentos. Sin embargo, debido a las limitaciones de hardware de la época, se construyó en 1986 con la llegada del CDROM [2].

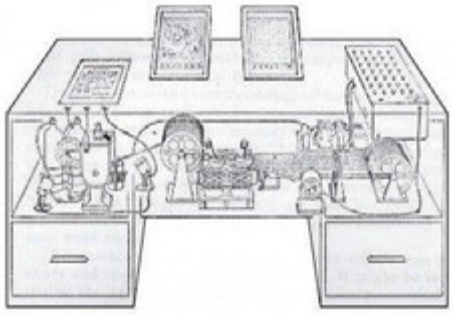


Fig.2 – Prototipo Memex en 1945 [2].

Fue en la década de los 50 cuando se incrementaron considerablemente los estudios para crear un ordenador que permitiera interaccionar con el usuario mediante una interfaz visual. En el año 1956 se creó el ordenador TX-0 con las operaciones de guardar, sumar y bifurcar. Debido al éxito de este ordenador, comenzaron con el avance de una versión mejorada, la TX-1, sin embargo, por su excesiva complejidad no se logró terminar el proyecto. Entonces en el año 1958, se creó la TX-2, máquina que inspiró a enormes estudiantes del momento.

Ivan Edward Sutherland es un científico de la computación estadounidense que tuvo acceso a la TX-2 durante su período de estudiante en ingeniería eléctrica. En el año 1963, Sutherland conocía la capacidad del ordenador para manipular imágenes a través de dibujos, lo que le llevó a diseñar para su tesis doctoral el famoso programa informático Sketchpad. Fue el primer programa informático capaz de modificar los elementos gráficos de manera sencilla.

Fig.3 – Ivan Sutherland con el Sketchpad [3].

Requería de un lápiz óptico para dibujar directamente en el monitor del ordenador, e incorporaba técnicas gráficas de interfaz de usuario como la capacidad de borrar líneas y el zoom [3].

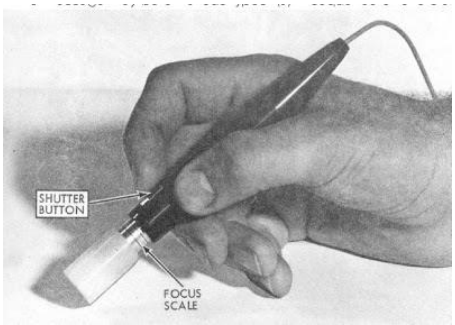


Fig.4 – Lápiz óptico del Sketchpad [3].

Con este sistema, las representaciones visuales pasaban a formar un objeto en sí mismo y no un gráfico. Gracias a este sistema se podía acceder al propio objeto para cambiar una propiedad concreta, mientras que anteriormente todo elemento formaba parte de un mismo gráfico, por tanto, no se podía acceder a unidades de elementos propias si no a la totalidad. A raíz de este sistema, se dio pie a la programación orientada a objetos.

Con el crecimiento de las interfaces gráficas, Douglas Engelbart en 1963 diseñó el primer prototipo del ratón para facilitar la interacción del usuario, su manejo y control se ha mantenido en esencia hasta la actualidad. Consistía en una pieza cómoda de controlar con la mano que permitía el movimiento horizontal y vertical del cursor del ratón, también contaba con un botón para hacer clic [4].

Fig.5 – Diseño del primer ratón [4].

La principal ventaja del ratón era que permitía su uso inmediato a cualquier tipo de persona, aunque no tuviera experiencia con el uso del ordenador.

En el año 1965, Sutherland se convirtió en profesor de la universidad de Utah y siguió con las investigaciones de computación gráfica. Ese mismo año presentó un programa de investigación sobre el grafismo computarizado para dar a conocer la necesidad que se tiene sobre una interfaz de usuario [5].

En el año 1968 Sutherland y David Evans fundaron la compañía “Evans & Sutherland”, con el objetivo de crear un nuevo hardware [6]. Fue entonces cuando inmensas corporaciones empezaron a investigar también sobre hardware para mejorar los gráficos de los ordenadores.

Este interés también formó parte en el mercado de las películas y animaciones. En 1969 se creó en la Universidad de Denver el sistema Scanimate, un sistema de animación analógica por ordenador. Su capacidad de crear animaciones en tiempo real desencadenó una gran repercusión en la publicidad, las promociones y las aperturas de programas. La National Film Board of Canada empezó a utilizar diversos métodos de animación y procedimientos de sombreado de interpolación [7].

Fig.6 – Película “Animation from Cape Dorset” en 1973 por la National Film Board of Canada [8].

En la década de los 70, se estaban estudiando en la universidad de Utah procedimientos para mostrar objetos tridimensionales. Entonces surgió el proyecto ARPA (Advanced Research Projects Agency) con novedades como el sombreado Gouraud, Phong y mapas de texturas [6].

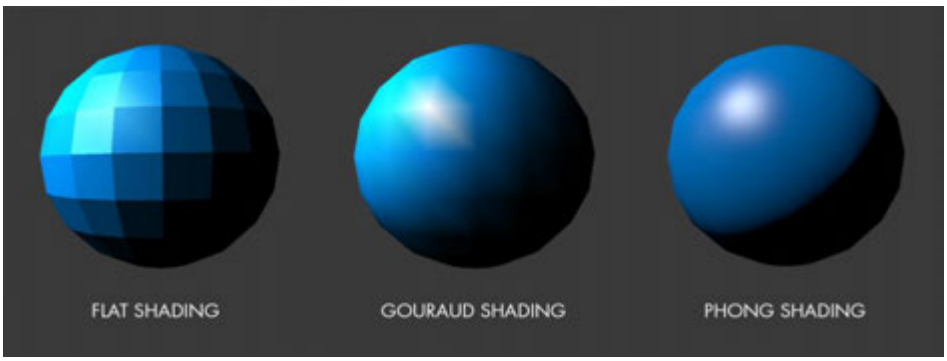


Fig.7 – Comparación de sombreados en esfera [9].

El sombreado de Phong obtiene mayor suavidad de sombras que el sombreado de Gouraud. Ya que, Phong permite calcular el color de cada píxel para ir reduciendo de forma degradada el color de cada uno y así, terminar con mejores acabados.

Todavía existían problemas de superposición y profundidad en una escena, sobre todo en decidir qué elementos son visibles y cuáles ocultos. Para solventar este problema, Ed Catmull desarrolló el "z-buffer" como un buffer capaz de almacenar las coordenadas de los elementos [10].

En 1971 la compañía Intel creó el primer microprocesador, el "4004". En seguida, comenzaron a desarrollarse mejores microprocesadores: en 1972 Intel lanza "8008", en 1974 National Semiconductor crea "SC/MP", en 1974 Intel presenta "8080", en 1975 Motorola crea "6800", en 1976 Zilog construye "Z80", en 1978 Intel lanza "8086" y "8088".

A partir de estas fechas, se realizaron continuamente nuevos avances. En 1972 se lanzó el videojuego Pong publicado por Atari. En 1975 Bill Gate y Paul Allen fundaron Microsoft. En 1976 Steve Wozniak y Steve Jobs lanzaron el primer Apple. En 1981 IBM creó los ordenadores personales. En 1983 Autodesk creó el software CAD para los ordenadores personales. Los ordenadores ya permitían crear interfaces gráficas [11].

En el mundo del cine, se puede destacar cortometrajes de Pixar con animaciones de computación gráfica 3D como Luxo Jr. (1986) El Sueño de Rojo (1987) y Tin Toy (1988). En 1995 Pixar recibe un premio de la Academia por la película "Toy Story", una de las primeras películas realizadas completamente por un ordenador [10].

2.2.3. Evolución de la realidad virtual

La realidad virtual es la emulación de un mundo tridimensional a través de sistemas tecnológicos que permite hacer sentir al usuario que forma parte de ese mundo.

Antes de la década los 50, existía el concepto de realidad virtual como una dimensión paralela accesible por el usuario. En el año 1935 Stanley G. Weinbaum (1902-1935) presentó el relato "Las gafas de Pigmalión", un relato sobre unas gafas capaces de envolver al portador de ellas con grabaciones holográficas, olor y tacto.

En la década de los 50 Morton Heilig presentó el prototipo “Sensorama”, un artefacto que proyectaba una filmografía y el espectador podía hacer uso de los cinco sentidos para mejorar su experiencia. Comenzó con 5 cortas filmografías [12].

Fig.8 – Sensorama en 1962 [13].

En el año 1960 Morton Heilig diseñó la “Máscara teleférica”, una pantalla portable que el usuario se debía poner en la cabeza para que con los ojos tuviera acceso a la pantalla y con los oídos a los altavoces que incorporaba [12].

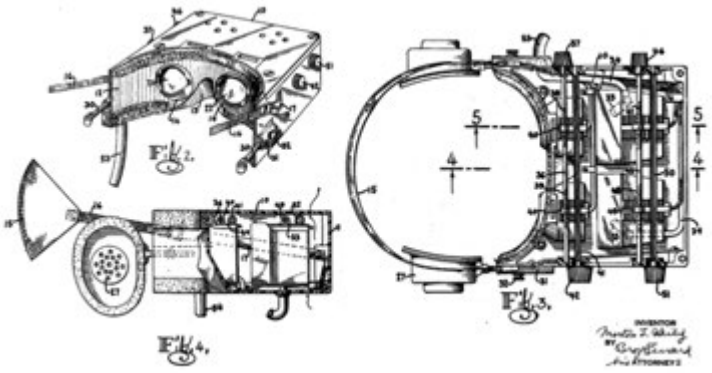


Fig.9 – Diseño de la máscara teleférica [14].

En el año 1961, se estaba buscando la mejor manera para los entrenamientos peligrosos de aterrizajes de los pilotos y las pruebas con artefactos explosivos y elementos químicos de los militares. Es por ello, por lo que Philco Corporation inventó el “Headsight”. Un dispositivo con una cámara para cada ojo y un sistema de detección de movimiento para simular el movimiento de cabeza del usuario [12].

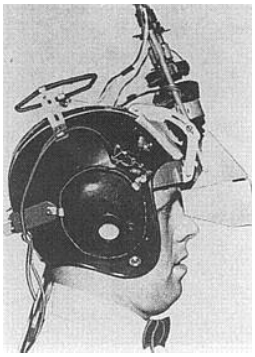


Fig.10 – “Headsight” en 1961 [15].

En el año 1968, Sutherland creó “La espada de Damocles”, un dispositivo de realidad virtual capaz de mostrar salas y objetos simples. Se empezaba a lograr un aspecto de realismo en la visión del espectador. Sin embargo, este aparato era demasiado pesado, por lo que se debía de fijar al techo para que el usuario no tuviera que soportar el peso en la cabeza. Por tanto, tenía la limitación de no ser portable [4].

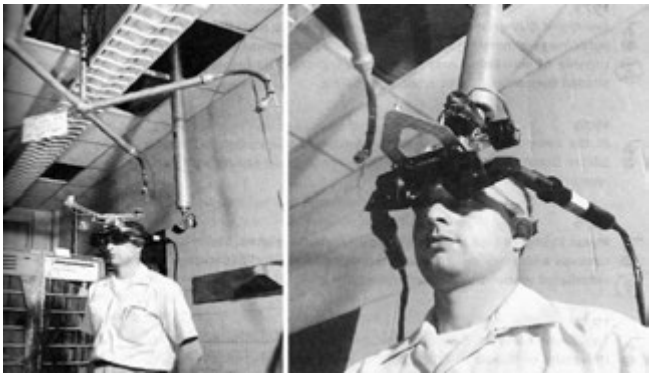


Fig.11 – “La espada de Damocles” en 1968 [16].

En los años 70 y 80, empezó a popularizarse el uso de la realidad virtual para las simulaciones de entrenamiento militar y marina, investigaciones químicas y conducción de vehículos terrestres y aéreos. A finales de los 80, la Fuerza Aérea Wright-Patterson creó una cabina virtual para adiestrar a los pilotos. Por otra parte, la NASA también vio potencial y en seguida se sumó a estas simulaciones de realidad virtual para preparar a los astronautas.

En los años 90 se dieron los primeros lanzamientos de realidad virtual con fines lúdicos. En 1991 Sega presentó la consola “Mega Drive” y las gafas de realidad virtual “Sega VR”, permitía detectar los movimientos de cabeza del usuario, tenía una cámara para cada ojo y auriculares. En 1995 Nintendo anunció la “Virtual Boy”, era una consola con forma de gafas, por tanto, las gafas pesarían bastante. Para soportar el peso, estas gafas tenían un atril de tal forma que se apoyaban en una mesa y el usuario debería ubicar su cabeza cerca de la mesa para ver por las gafas [12].

Fig.12 – Videoconsolas “Sega VR” [17], “Mega Drive” [18], “Virtual Boy” [19].

Ambas consolas fracasaron debido a las incomodidades físicas a las que se tenía exponer el usuario para jugar. En muchos casos acabaron con dolores de espalda y de cuello, además de problemas de salud como mareos, jaquecas, fuertes dolores de cabeza…

Estos fracasos provocaron que el mercado de videojuegos se alejase de la realidad virtual durante los años del 2000.

A partir del año 2010, se retomó el negocio de las gafas de realidad virtual. El principal impulsor fue Palmer Luckey con su arquetipo de Oculus Rift. Un visor con una pantalla de pequeñas pulgadas para cada lente, pero capaz de mostrar resoluciones HD. La primera versión se lanzó en 2012 con "Developer Kit" y la segunda en 2014 con "Developer Kit 2".

Fue entonces cuando empezaron a aparecer nuevas marcas de gafas de realidad virtual. En 2014 Valve en colaboración con HTC realizaron las gafas "Vive". A diferencia de las Oculus, las "Vive" se centrarían en la "escala de habitación", para que los usuarios puedan desplazarse por su propia habitación y naveguen por el mundo virtual. Ese mismo año, Sony también anunció su proyecto de innovación como realidad virtual y en el año 2016 salieron a la venta "Vive" y "PlayStation VR" [12].

Actualmente los visores de realidad virtual son muy aceptados en el mercado con millones de ventas y continúan desarrollándose nuevos.

2.2.4. Motores Gráficos

Un motor gráfico es una tecnología software que permite desarrollar videojuegos para múltiples plataformas como consolas, móviles u ordenadores [20].

Se puede diferenciar dos tipos de motor gráfico:

- Juegos en primera persona: tienen mayor complejidad a nivel de controles, puesto que el jugador debe controlar el movimiento y la visión del personaje. El personaje se encuentra en espacios tridimensionales donde puede interacciones con objetos y bots y en algunos casos teniendo que generar inteligencia artificial muy técnica para estos bots. Suelen disponer de modo multijugador online [20].
- Juegos de plataforma y de tercera persona: presentan menor complejidad a nivel de controles, ya que, el jugador debe controlar el movimiento del personaje y generalmente la cámara sigue al propio personaje del juego. También permiten mover la cámara, pero son desplazamientos más sencillos que los de primera persona. Los juegos de plataformas se incluyen en esta sección porque en la última generación traen muchos elementos tridimensionales añadiendo complejidad al juego [20].

Entre los componentes principales más destacables de un motor de videojuegos son:

- Programa de juego principal: Cada motor gráfico utilizada un lenguaje de programación principal diferente [21].
- Renderización: procedimiento para producir los distintos elementos de un archivo de escena: texturas, sombras, iluminaciones… Los polígonos son elementos renderizados tridimensionales y pueden tener distinto grado de detalle, típicamente son bajo, medio y alto de escala de detalle [21].
- Audio: un videojuego hoy en día debe tener música acompañada. La técnica de ajuste de audio más utilizada por los motores gráficos es el looping. Esta técnica consiste en reproducir el mismo fragmento de audio durante infinito tiempo, generalmente se ha estudiado un encaje musical entre el final del audio y el inicio [21].

A continuación, se va a mostrar una tabla comparativa de los 3 motores gráficos más utilizados:

TABLA 2.1. COMPARATIVA DE MOTORES GRÁFICOS

Carácte-rísticas	Unity	Unreal Engine 4	CryEngine 5
Plataformas	PS4, Xbox One, Xbox 360, SteamOS, Windows, Phone 8, Tizen, Android TV y Samsung SMART TV, WebGL, Windows, Mac OS X, Linux, Web Player, Playstation Vita, Wii U, iOS, Android y VR (incluido Hololens) [22].	iOS, Android, VR, Linux, PC con Windows, Mac OS X, SteamOS, HTML5, Xbox One y PS4 [22].	Windows, Linux, PlayStation 4, Xbox One, Oculus Rift, OSVR, PSVR y HTC Vive. El soporte móvil está en desarrollo [23].
Codificación	CSharp y Javascript, aunque poco a poco se está alejando del Javascript [22].	C++ [24].	C++ [25].
Complementos	Dispone de la "asset store", una tienda online muy completa con escenarios, armas, sonidos, módulos de control… Gran cantidad de estos assets son gratuitos [26].	Dispone de "Unreal Engine marketplace", una tienda online con animaciones, sonidos y materiales 3D gratuitos y de pago [26].	Dispone de "CryEngine marketplace", una tienda online con cientos de materiales, sonidos y objetos 3D con precios variados, algunos incluso son gratuitos [23].
Soporte	Mantiene soporte 2D y 3D [27].	Muy buena para juegos 3D y realidad virtual. Problemas con juegos retro o en 2D [28].	Alta potencia para juegos 3D con estereoscópico [27].
Rendimiento	En modo edición del proyecto, tiene varias tareas activas, por consiguiente, se reduce el rendimiento y el ordenador se calienta [29].	En modo edición y compilación la CPU se dispara. Pero mantiene tiempos cortos para la compilación [30].	En modo de edición y compilación consume muchos recursos [27].
Almacenamiento	Los proyectos aumentan su tamaño rápidamente si se usan assets complejos, escenas cargadas de elementos, modelados con buena resolución [29].	Cualquier proyecto ocupa mucho espacio, incluso los proyectos de juegos pequeños ocupan bastante espacio en el disco duro [27].	Debido a los acabados que tiene cualquier escenario 3D, el espacio de almacenamiento es elevado. Resulta complicado mantener proyectos con poco espacio [31].
Interfaz	Simple y muy intuitiva. Permite arrastrar y soltar elementos para referenciar [32].	Permite la opción de programación basada en nodos y componentes para facilitar el trabajo a los nuevos usuarios [33]	Cuenta con una interfaz intuitiva para usuarios experimentados en los motores gráficos [34]. Actualmente está trabajando en iconos y ventanas.
Precio	Cuenta con una versión gratuita, una versión Unity Pro por 150 dólares al mes y otra versión Unity Plus por 40 dólares al mes [35].	Cuenta con una versión de pago de 19 dólares al mes, o bien pagar a la compañía Epic un 5% de los primeros 3000$ de beneficios del proyecto [24].	Adopta un modelo de negocio "paga lo que quieras". Por tanto, el precio por la suscripción lo define el propio usuario [36].

Además, los 3 motores gráficos cuentan con tutoriales de ayuda al iniciar la experiencia del usuario para evitar su confusión y facilitarle la comprensión de la interfaz

del usuario. Estos motores ofrecen una interfaz con la ubicación de las ventanas de trabajo por defecto, pero el usuario puede reubicarlas a su gusto para trabajar con mayor comodidad.

Un punto fuerte que tienen estos motores es que gracias a la fama que han alcanzado, existen gran cantidad de foros de ayuda por internet. De manera que los usuarios novatos pueden recabar mucha información para sus primeros proyectos.

2.2.5. Línea del tiempo de la industria del videojuego

Década de los 50: El 90% de los norteamericanos tenían al menos una televisión. Por tanto, se empezaron a investigar mayores provechos a las televisiones [37].

- **1958**: William Higginbotham desarrolló el videojuego "Tennis for Two" para simular partidas de tenis con un osciloscopio como pantalla. El juego estaba formado por una línea horizontal para indicar el suelo y una línea vertical para señalar la red. Los jugadores podían devolver la bola eligiendo un valor de ángulo [38].

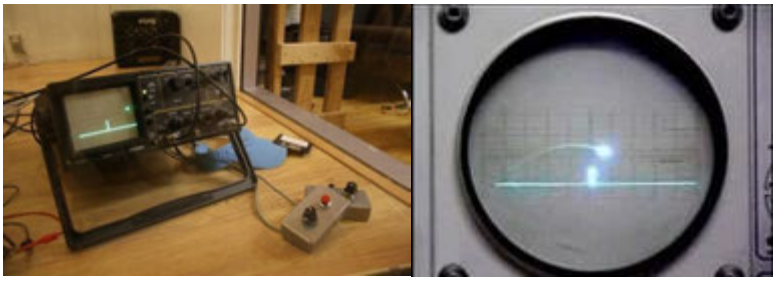


Fig.13 – Videojuego "Tennis for Two" en 1958 [39].

Década de los 60: Época de numerosos fracasos de proyectos de desarrollo de videojuegos.

- **1962**: Steve Russell, Wayne Witaenem y Martin Graetz desarrollaron el videojuego "Spacewar!". Este juego consistía en una batalla entre dos naves en el espacio exterior, las naves tendrían limitaciones de misiles y combustible. Además, incorporaba la pseudoaleatoriedad con la aparición de enemigos en posiciones aleatorias [38].

Fig.14 – Videojuego "Spacewar!" en 1962 [40].

Década de los 70: Aparecen las máquinas recreativas y las videoconsolas. El negocio de los videojuegos comienza a explotarse. A partir de esta década, se van a mencionar los juegos más novedosos y revolucionarios de la época, ya que, también se crearon muchos otros juegos también exitosos pero que eran clones de los juegos

revolucionarios. Por tanto, muchas empresas se lucraban de las innovaciones de otras, vendiendo sus consolas y juegos a menor precio.

- **1971**: Bill Pitts y Hugh Tuck llevan el exitoso juego “Spacewar!” a la primera máquina de recreativos bajo el nombre de “Galaxy Game”. Nolan Bushnell y Ted Dabney crean el juego “Computer Space” donde el jugador controla un cohete que puede disparar a los platillos voladores [37].
- **1972**: Nolan Bushnell y Ted Dabney fundan la empresa Atari. Ralph Baer desarrolla la primera videoconsola “Magnavox Odyssey” con el videojuego “Pong” que simula el tenis de mesa. El jugador controla la raqueta izquierda con movimientos verticales y el adversario el lado opuesto [41].
- **1973**: Atari crea “Space Race” y “Gotcha”.
- **1974**: Atari lanza “Rebound”, “Quadrapong”, “Touch-Me”, “Tank”, “Qwak!” y “Gran Tak 10”.
- **1975**: Atari crea la segunda videoconsola de la historia “Tele-Games Pong”. Taito presenta la recreativa “Gun Fight”, fue el introductor del género de lucha y de la separación de los controles de movimiento y dirección. La compañía Exidy muestra “Destruction Derby”.
- **1976**: Atari lanza “Breakout”. Exidy crea “Death Race”. Fairchild Semiconductor crea la consola “The Fairchild Channel F”. Mattel Electronics presenta “Auto Race” el primer videojuego portátil. “Warner Bros” compra Atari por 28 millones de dólares.
- **1977**: RCA lanza la videoconsola “RCA Studio II”. Mattel Electronics crea otro videojuego portátil “Football”.
- **1978**: Taito lanza “Space Invaders”, un juego inspirado en la famosa película “Star Wars” y que logró ganarse la atención de todos sus seguidores.
- **1979**: Atari crea “Asteroids” y “Lunar Lander”. Namco lanza “Galaxian” [37].

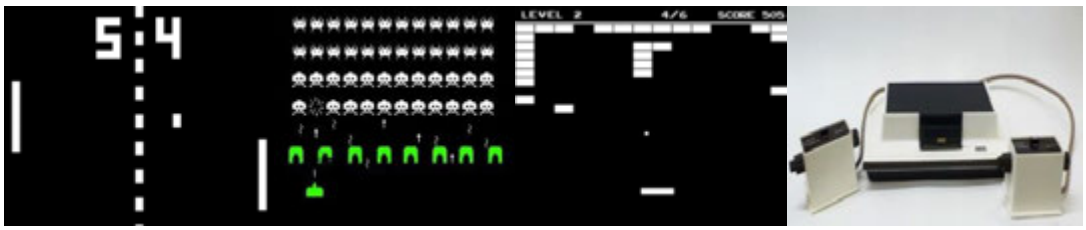


Fig.15 – “Pong” [42], “Space Invaders” [43], “Breakout” [44] y “Magnavox Odyssey” [45].

<u>Década de los 80</u>: Los videojuegos pasan de ser un entretenimiento a ser un gran avance tecnológico, se da paso a la época dorada de los videojuegos con los ordenadores de 8 bits.

- **1980**: Atari lanza “Missile Comand” que seguía la misma línea de juego de naves espaciales hasta el momento [41]. Namco crea “Pac-Man” muy revolucionario que alcanzó millones de ventas donde el jugador deberá conseguir todas las bolas de puntos en un laberinto con cuatro fantasmas enemigos [46].
- **1981**: Atari lanza “Tempest” y “Centipede”. Williams Electronics crea “Defender”. Namco lanza “Galaga”. Sega saca “Frogger” [37]. Nintendo lanza “Donkey Kong” [67]. Juegos muy aceptados por los jugadores.
- **1982**: Atari crea “Space Duel”. Sega lanza “Football Manager”, un juego de ordenador muy popular por ser de los primeros juegos de estrategia de fútbol. A

partir de este año, los juegos no solo se centran en máquinas recreativas si no que dan el salto a la plataforma del ordenador [47].

- **1983**: Matthew Smith desarrolla el videojuego “Manic Miner” para ordenador, un juego de plataformas totalmente novedoso con color y música acompañada [48]. La compañía Paco&Paco lanza “La Pulga”, es considerado el primer videojuego original desarrollado en España. Nintendo crea la consola “Nintendo Entertainment System” alcanzó un gran éxito y vino acompañada con el juego “Donkey Kong” [49].
- **1984**: Matthew Smith continúa con el desarrollo de videojuegos de plataformas y lanza “Jet Set Willy” con enemigos variados, controles minimalistas y escenarios muy vivos [37]. Ultimate Play The Game presentan “Knight Lore”, un juego sencillo de plataformas, pero con la novedad de movimientos e interacciones con objetos en perspectiva isométrica 3D [50]. Konami lanza “Hyper Sports”. Nintendo lanza “Tennis” [46]. Alekséi Pázhitnov creó el famoso videojuego “Tetris”, un juego donde unas figuras 2D van cayendo de uno en uno hasta posicionarse en suelo con el objetivo de formar líneas completas horizontales, tuvo grandísima repercusión para el resto de las compañías y pronto empezaron a crearse otras copias del mismo juego [51].
- **1985**: Ultimate Play The Game lanza “Alien 8” y “Nightshade”. Konami crea “Yie Ar Kung-Fu”. Capcom expone “Commando” y “Ghosts 'n Goblins”. Epyx presenta “Winter Games” [37]. Nintendo lanza “Super Mario Bros”, el primer juego de la franquicia de Mario, un juego de plataformas acompañado de enemigos, música y ambientes nunca vistos alcanzó tan nivel de popularidad que se convirtió en el icono principal de Nintendo [52]. Sega crea la videoconsola “Master System”, una consola que no superó las ventas de las principales del momento, pero que permitió generar suficientes beneficios para sentar las bases de Sega [53].
- **1986**: Se continuaron desarrollando juegos que mezclaban la isometría del 3D y plataformas 2D. Ultimate Play the Game crea “Gunfright”. Ocean Software lanza “Batman”. Hewson Consultants presenta “Uridium”. Dinamic Software expone “Camelot Warriors”. Nintendo lanza en Europa “The Legend of Zelda” [49].
- **1987**: Ultimate Play the Game lanza “Head Over Heels”. Taito crea “Operation Wolf”. Hewson Consultants lanza “Ranarama” y “Nebulus”. Dinamic Software presenta “Basket Master” [37]. Capcom lanza “Street Fighter”, un juego arcade de peleas que alcanzó cierta popularidad, lo que impulsó a la compañía a continuar con esa saga que acababa de nacer [54]. Square Enix presenta “Final Fantasy”, un juego que salvó de la bancarrota a la compañía debido a la innovación que traía, por ser un juego con una historia diferente al resto, donde los personajes cobraban importancia y vivían en unos escenarios muy variados [55].
- **1988**: Nintendo lanzó “Super Mario Bros 2” y “Super Mario Bros 3”, los últimos juegos de la famosa consola “Nintendo Entertainment System”. Square Enix lanza “Final Fantasy 2” [37].
- **1989**: Nintendo aparece con la “Game Boy”, una videoconsola portátil de 8 bits. Inmediatamente Nintendo se posiciona como líder en ventas de las consolas portátiles [49].

Fig.16 – Principales consolas del mercado de la década de los 80: “Sega Master System” [56] , “Nintendo Entertainment” [57] y “Game Boy” [58].

Fig.17 – Juegos referentes de la década de los 80: “Pac-Man” [59], “Knight Lore” [60] , “Super Mario Bros” [61] , “Street Fighter” [62] y “Tetris” [63].

Década de los 90: Época de desarrollo de videojuegos más avanzados. Las principales categorías eran juegos de acción, juegos narrativos y juegos de simulación y estrategia.

- **1990**: LucasFilm Games lanza “Loom” y “The Secret of Monkey Island”, fueron calificados como videoaventuras excelentes gracias a su motor gráfico [37]. Nintendo crea la “Super Nintendo” con poco repertorio de videojuegos [64].
- **1991**: La plataforma cinemática de los videojuegos estaba en completo auge, lo que llevó a que más compañías también se sumaran al carro, como Delphine Software que lanza “Another World” o ID Software con “Catacomb 3-D” [37]. Por otra parte, MicroProse presenta “Civilization”, un videojuego de estrategia por turnos que consiste en formar ciudades, construir, comandar y dirigir ejércitos hasta conquistar el planeta, en 1992 fue galardonado con el premio Origins al mejor videojuego de estrategia del año [61].
- **1992**: Infogrames crea “Alone in the Dark”, cuenta con gráficos 3D poligonales y fue precursor del género survival horror [65]. ID Software lanza “Wolfenstein 3D” de disparos en primera persona [37]. Nintendo anuncia “Super Mario Kart” con millones de ventas [66].
- **1993**: Adventure Soft presenta “Simon the Sorcerer”, un juego 2D, pero con fama debido a que se basaba en una parodia de numerosos cuentos de hadas y películas conocidas del momento [67]. ID Software crea “Doom” un juego de disparos famoso por la cantidad de violencia explícita que mostraba, también popularizó el juego en red [65].
- **1994**: Blizzard Entertainment lanza “Warcraft: Orcs & Humans”, precursor de los juegos de estrategia en tiempo real en escenarios de estilo medieval donde se enfrentan humanos contra diversas criaturas fantásticas. Raven Software lanza “ShadowCaster” [66]. Sega crea la consola “Sega Saturn” de 32 bits, famosa por su juego de peleas “Virtua Fighter” [37]. Sony presenta “Play Station” de 32 bits que rápidamente se posicionó como la consola más vendida debido a su amplio catálogo y al formato de venta de juegos en CD [68].

- **1995**: LookingGlass Technologies crea “Flight Unlimited”, un simulador de vuelos con avión que incluía imágenes reales. Nintendo anuncia la consola “Virtual Boy”, la cual tuvo un fracaso en ventas [37].
- **1996**: Nintendo crea la nueva consola “Nintendo 64”, de 64 bits. Esta consola tenía tecnología superior a la de sus rivales, pero el formato de venta de juegos era por cartuchos, lo que encarecía el precio [49]. Por tanto, durante esta década se mantuvo en primer puesto Sony.
- **1997**: Microsoft Game Studios lanza “Age of Empires”, un juego de estrategia en tiempo real [69].
- **1998**: Pyro Studios lanza “Commandos: Behind Enemy Lines” de estrategia en tiempo real. Looking Glass Studios crea “Thief: The Dark Project”, primer juego de sigilo en primera persona [37].

Fig.18 – Principales consolas del mercado de la década de los 90: “Nintendo 64” [70], “Play Station” [71] y “Sega Saturn” [72].

Fig.19 – Juegos referentes de la década de los 90: “The Secret of Monkey Island” [73], “Civilization” [74], “Doom” [75] y “Super Mario 64” [76].

Del 2000 hasta hoy: La industria de las máquinas de recreativos se quedó obsoleta siendo sustituida por las videoconsolas. Los jugadores tenían la posibilidad de jugar en sus hogares con una tecnología similar o superior a las recreativas perdiendo totalmente el interés en ellas.

- **2000** Sony presentó la “Play Station 2” de 128 bits convirtiéndose en la consola más vendida de la historia [77]. En **2001** Microsoft anunció la “X-Box” [78] y Nintendo creó la “GameCube” [49] pero ninguno consiguió igualar el éxito de Sony [68]. Nintendo también probó el mercado de las consolas portátiles con la “Game Boy Advance” [49].
- **2004** Nintendo dio a luz la “Nintendo DS”, un éxito de ventas en consolas portátiles con la novedad de una pantalla táctil y posibilidad de estado de suspensión [49]. Ese mismo año, Sony anunció “Play Station Portable”, una consola portátil que no llegó a la línea de Nintendo [68].
- **2005** Microsoft lanzó “Xbox 360” para competir contra la “Play Station 2” [71], la respuesta de Sony fue en 2006 con la llegada de la “Play Station 3” [68]. Pero la novedad vino con Nintendo en 2006 con la “Wii”, que tenía un innovador sistema de control por movimiento [49].
- **2007** Steve Jobs anunció el iPhone, un móvil con pantalla táctil y de tecnología elevada que soportaba la transmisión de datos 3G a altas velocidades. Esto

propulsó el desarrollo de los teléfonos móviles táctiles, empezando así el mercado de los videojuegos para móviles [79].

- **2010** Steve Jobs presentó el "iPad", un dispositivo táctil que actuaría como un ordenador personal, pero de forma ligera y portable. También tendría la compatibilidad con conexión wi-fi, 3G y GPS [80]. Por otra parte, Microsoft anunció el "Kinect", una cámara que actuaría como complementario a su consola "Xbox 360" para detectar el movimiento de los jugadores y así mantener cierto parecido con la "Wii" de Nintendo [37].
- **2012** King anunció el "Candy Crush Saga" y Supercell el "Clash Of Clans", dos juegos gratuitos, pero con formas de pago para acelerar los progresos del juego y, así logró enormes éxitos [37]. Actualmente, las consolas más conocidas y que siguen en el mercado son "Play Station 4" anunciada en el **2014** por Sony [68], "Xbox One" lanzada por Microsoft en **2014** y "Nintendo Switch" presentada por Nintendo en **2016** [37].
- **2016** Supercell lanzó "Clash Royale", un juego de móvil de estrategia con cartas y Nintendo anunció "Pokémon Go", un juego que utiliza la ubicación gps para hacer mover al jugador por el mundo real en busca de aventuras [37]. También salió a la venta las gafas de realidad virtual de "Play Station 4" que permiten compatibilidad con inmensa cantidad de juegos ya existentes [68].

Fig.20 – Principales consolas del mercado actual: "Nintendo Switch" [81], "Play Station 4" [82] y "Xbox One" [83].

Fig.21 – Juegos conocidos actualmente: "Smash Brosh Ultimate" [84], "Fortnite" [85], "Atro Bot VR" [86] y "Clash Royale" [87].

2.2.6. Conclusiones

En conclusión, se ha podido observar que el mercado de los videojuegos es algo que se ha mantenido en explotación desde la década de los 70 y que no parece que vaya a desaparecer. Los videojuegos continúan generando beneficios independientemente del estado económico en el que se encuentre España. Si no hubiera crisis económica los beneficios serían mayores, sin embargo, debido a la variedad de precios que puede presentar un videojuego y el soporte multiplataforma, facilita a que el usuario disponga de la tecnología necesaria para poder ejecutar el videojuego, por consiguiente, mayor probabilidad de venta.

Se ha observado que cada década marca una novedad haciendo que en la siguiente década se saque el máximo provecho de la innovación. En esta última década lo más novedoso son los videojuegos móviles y de realidad virtual. Además, se ha visto que la realidad virtual desde su origen ha presentado un enorme potencial en aspectos didácticos y de simulación. Por tanto, aprovechando el apogeo de la realidad virtual, su capacidad de enseñanza y su elevada aptitud para la rehabilitación de pacientes reales, se ha decidido implementar el juego en realidad virtual.

Al tratarse de realidad virtual, el videojuego tiene que formarse alrededor de entornos 3D. La toma de decisión del motor gráfico resulta complicada sabiendo que los tres motores explicados anteriormente son potenciales en juegos 3D.

Un aspecto fundamental en este proyecto es el escaso nivel de experiencia del alumno en el desarrollo de los videojuegos, Unity y Unreal Engine 4 son populares por la cantidad de tutoriales, foros de ayuda, diseño de interfaz que facilitan el aprendizaje de los usuarios novatos, sin embargo, Cry Engine da por entendido muchos aspectos, lo que implica que omite iconos y elementos principales que pueden hacer que un usuario novato se confunda y dificulte su progreso. Por tanto, Cry Engine se descarta en la decisión.

Unity y Unreal Engine 4 son capaces de generar entornos 3D con alta capacidad de efectos gráficos, buena iluminación y texturas realistas. Con ambos se va a ocupar mucho espacio de almacenamiento para los proyectos, por lo que este factor no sirve de ayuda para la toma de decisión. Sin embargo, Unreal Engine 4 se conoce por consumir más CPU que Unity porque permite una calidad gráfica mayor que la de Unity y además tiene una comunidad inferior a la de Unity porque este motor se creó años más tarde.

Hay que tener en cuenta que implementar juegos en realidad virtual ya consume bastante CPU por sí mismo, si a esto se le agrega el consumo de Unreal Engine 4, será más difícil poder compilar y probar el juego. Por tanto, ese motive sirve para descartar el motor Unreal Engine y se hará uso del motor gráfico Unity.

2.3. Requisitos

El propósito de esta sección es exponer los requisitos que deberá cumplir el prototipo final del videojuego.

2.3.1. Definición de requisitos

Para la definición de los requisitos se va a seguir el siguiente esquema:

TABLA 2.2. DEFINICIÓN DE REQUISITO

Identificador	
Nombre	
Prioridad	
Necesidad	
Descripción	

- Identificador: etiqueta única para distinguir cada requisito.
- Nombre: presentación resumida del requisito
- Prioridad: preferencia que se mantiene al requisito. Se utilizarán 3 niveles:
  - Alta
  - Media
  - Baja
- Necesidad: obligación del requisito. Se utilizarán 3 grados:
  - Esencial
  - Deseable
  - Opcional
- Descripción: explicación del requisito.

2.3.2. Requisitos funcionales iniciales

Los requisitos funcionales definen las funciones que debe cumplir el sistema y su respuesta ante diferentes comportamientos y patrones tanto de entrada como de salida [88].

Inicialmente, los requisitos funcionales son los siguientes.

TABLA 2.3. REQUISITO FUNCIONAL - MOVER MANOS

Identificador	**RF01**
Nombre	Mover manos
Prioridad	Alta
Necesidad	Esencial
Descripción	El usuario podrá mover las manos desplazando los mandos.

TABLA 2.4. REQUISITO FUNCIONAL. MOVER CABEZA

Identificador	**RF02**
Nombre	Mover cabeza
Prioridad	Alta
Necesidad	Esencial
Descripción	El usuario podrá observar el entorno virtual girando la cabeza.

TABLA 2.5. REQUISITO FUNCIONAL - DESPLAZAMIENTO DEL JUGADOR

Identificador	**RF03**
Nombre	Desplazamiento del jugador
Prioridad	Alta
Necesidad	Esencial
Descripción	El usuario podrá desplazarse por el entorno virtual moviéndose por el entorno real.

TABLA 2.6. REQUISITO FUNCIONAL – AGACHARSE

Identificador	**RF04**
Nombre	Agacharse
Prioridad	Alta
Necesidad	Esencial
Descripción	El usuario podrá agacharse en el entorno virtual agachándose en el entorno real.

TABLA 2.7. REQUISITO FUNCIONAL – SALTO

Identificador	**RF05**
Nombre	Salto
Prioridad	Alta
Necesidad	Esencial
Descripción	El usuario podrá saltar en el entorno virtual saltando en el entorno real.

TABLA 2.8. REQUISITO FUNCIONAL – DESTRUCCIÓN DE OBSTÁCULO

Identificador	**RF06**
Nombre	Destrucción de obstáculo
Prioridad	Alta
Necesidad	Esencial
Descripción	El usuario podrá destruir el obstáculo tocando con la mano el obstáculo.

TABLA 2.9. REQUISITO FUNCIONAL – APARICIÓN DE OBSTÁCULO

Identificador	**RF07**
Nombre	Aparición de obstáculo
Prioridad	Alta
Necesidad	Esencial
Descripción	Cada obstáculo deberá aparecer tras un intervalo concreto de tiempo.

TABLA 2.10. REQUISITO FUNCIONAL – SONIDO DE DESTRUCCIÓN

Identificador	**RF08**
Nombre	Sonido de destrucción
Prioridad	Baja
Necesidad	Opcional
Descripción	El sistema debe reproducir un sonido cada vez que un obstáculo es destruido.

TABLA 2.11. REQUISITO FUNCIONAL – REPRODUCCIÓN DE MÚSICA

Identificador	**RF09**
Nombre	Reproducción de música
Prioridad	Baja
Necesidad	Opcional
Descripción	El sistema debe reproducir música de ambiente.

TABLA 2.12. ELECCIÓN DE DURACIÓN

Identificador	**RF10**
Nombre	Elección de duración
Prioridad	Alta
Necesidad	Esencial
Descripción	El sistema deberá permitir elegir una duración total de la partida.

TABLA 2.13. REQUISITO FUNCIONAL – FIN DE LA PARTIDA

Identificador	**RF11**
Nombre	Fin de la partida
Prioridad	Alta
Necesidad	Esencial
Descripción	El sistema deberá finalizar la partida tras haber terminado el tiempo de duración total de partida.

TABLA 2.14. REQUISITO FUNCIONAL – OBSTÁCULOS DELANTEROS

Identificador	**RF12**
Nombre	Obstáculos delanteros
Prioridad	Alta
Necesidad	Esencial
Descripción	El sistema no deberá tener en cuenta en los resultados, los obstáculos que situadas por delante del jugador al finalizar la partida.

TABLA 2.15. REQUISITO FUNCIONAL – FICHERO DE SALIDA

Identificador	**RF13**
Nombre	Fichero de salida
Prioridad	Alta
Necesidad	Esencial
Descripción	El sistema deberá escribir en un fichero salida “txt” la fecha en la que se realiza la partida, la duración de la partida, la cantidad total de obstáculos y el total de precisión del jugador con los obstáculos.

2.3.3. Requisitos funcionales adicionales

Se han agregado requisitos funcionales adicionales sobre los originales a raíz del desarrollo de las alternativas de diseño explicadas en la sección “5. Diseño de la solución técnica”.

TABLA 2.16. REQUISITO FUNCIONAL – DESPLAZAMIENTO DEL OBSTÁCULO

Identificador	**RF14**
Nombre	Desplazamiento del obstáculo
Prioridad	Alta
Necesidad	Esencial obstáculo
Descripción	Todos los obstáculos deberán desplazarse en dirección paralela al suelo.

TABLA 2.17. REQUISITO FUNCIONAL – SENTIDO DEL OBSTÁCULO

Identificador	**RF15**
Nombre	Sentido del obstáculo
Prioridad	Alta
Necesidad	Esencial
Descripción	Todos los obstáculos deberán desplazarse en el sentido que se aproximen a la posición inicial del jugador.

TABLA 2.18. REQUISITO FUNCIONAL – VELOCIDAD DEL OBSTÁCULO

Identificador	**RF16**
Nombre	Velocidad del obstáculo
Prioridad	Alta
Necesidad	Esencial
Descripción	Todos los obstáculos deberán desplazarse a una velocidad concreta.

TABLA 2.19. REQUISITO FUNCIONAL – ATRAVESAMIENTO DEL OBSTÁCULO

Identificador	**RF17**
Nombre	Atravesamiento del obstáculo
Prioridad	Alta
Necesidad	Esencial
Descripción	Todos los obstáculos situados por detrás de las manos del jugador deberán ser capaces de atravesar al jugador.

TABLA 2.20. REQUISITO FUNCIONAL – ELECCIÓN DE VELOCIDAD

Identificador	**RF18**
Nombre	Elección de velocidad
Prioridad	Alta
Necesidad	Esencial
Descripción	El sistema deberá permitir elegir una velocidad de desplazamiento de los obstáculos.

TABLA 2.21. REQUISITO FUNCIONAL – ELECCIÓN DEL INTERVALO

Identificador	**RF19**
Nombre	Elección del intervalo
Prioridad	Alta
Necesidad	Esencial
Descripción	El sistema deberá permitir elegir un intervalo de tiempo de aparición entre cada obstáculo.

Se ha decidido ampliar el requisito “RF13” para aumentar la cantidad de datos que se escribirán en el fichero de salida. Por tanto, el requisito funcional “RF13” será el siguiente:

TABLA 2.22. REQUISITO FUNCIONAL – FICHERO DE SALIDA AMPLIADO

Identificador	**RF13**
Nombre	Fichero de salida
Prioridad	Alta
Necesidad	Esencial
Descripción	El sistema deberá escribir en un fichero salida “txt” la fecha en la que se realiza la partida, la duración de la partida, la velocidad de movimiento de los obstáculo, el intervalo de tiempo de salida entre cada obstáculo, la cantidad total de obstáculos de cada color y totales, la cantidad de obstáculos golpeados de cada color y totales, la cantidad de obstáculos no golpeados de cada color y totales, la cantidad de obstáculos golpeados incorrectamente de cada color y totales y el total de precisión del jugador con los obstáculos de cada color y totales.

2.3.4. Requisitos no funcionales

Los requisitos no funcionales definen las restricciones para la elaboración del sistema [88] . Estos requisitos se van a clasificar de acuerdo con el criterio que ocupan.

2.3.4.1. Requisitos no funcionales de rendimiento

TABLA 2.23. REQUISITO NO FUNCIONAL – FLUIDEZ DEL JUEGO

Identificador	**RNF01**
Nombre	Fluidez del juego
Prioridad	Alta
Necesidad	Esencial
Descripción	El videojuego deberá mostrarse fluidamente sin ralentizaciones ni paradas.

TABLA 2.24. REQUISITO NO FUNCIONAL – FLUIDEZ DE LA MÚSICA

Identificador	**RNF02**
Nombre	Fluidez de la música
Prioridad	Media
Necesidad	Opcional
Descripción	La música deberá ser capaz de escucharse de forma clara.

2.3.4.2. Requisitos no funcionales de portabilidad

TABLA 2.25. REQUISITO NO FUNCIONAL – SISTEMAS OPERATIVOS

Identificador	**RNF03**
Nombre	Sistemas operativos
Prioridad	Alta
Necesidad	Esencial
Descripción	El proyecto creado para el videojuego no debe contener elementos que impidan su funcionalidad en los sistemas operativos Windows y Macintosh.

2.3.4.3. Requisitos no funcionales de estabilidad

TABLA 2.26. REQUISITO NO FUNCIONAL – ERRORES DEL VIDEOJUEGO

Identificador	**RNF04**
Nombre	Errores del videojuego
Prioridad	Alta
Necesidad	Esencial
Descripción	El videojuego no debe generar ningún error que obligue al cierre inmediato.

2.3.4.4. Requisitos no funcionales de accesibilidad

TABLA 2.27. REQUISITO NO FUNCIONAL – EDAD DEL USUARIO

Identificador	**RNF05**
Nombre	Edad del usuario
Prioridad	Alta
Necesidad	Esencial
Descripción	El videojuego debe ser jugable para cualquier usuario independientemente de su edad.

TABLA 2.28. REQUISITO NO FUNCIONAL – IDIOMA DEL USUARIO

Identificador	**RNF06**
Nombre	Idioma del usuario
Prioridad	Media
Necesidad	Opcional
Descripción	El videojuego debe ser jugable para cualquier tipo de usuario independientemente de su idioma natal.

2.3.4.5. Requisitos no funcionales de usabilidad

TABLA 2.29. REQUISITO NO FUNCIONAL – FÁCIL DE USAR

Identificador	**RNF07**
Nombre	Fácil de usar
Prioridad	Alta
Necesidad	Esencial
Descripción	El videojuego debe ser fácil de usar para todos los jugadores independientemente de su experiencia en los videojuegos.

TABLA 2.30. REQUISITO NO FUNCIONAL – EFICIENCIA

Identificador	**RNF08**
Nombre	Eficiencia
Prioridad	Alta
Necesidad	Esencial
Descripción	El videojuego debe responder con un retraso máximo de medio segundo de las acciones del usuario.

2.3.4.6. Requisitos no funcionales de interfaz

TABLA 2.31. REQUISITO NO FUNCIONAL – INTUITIVO

Identificador	**RNF09**
Nombre	Intuitivo
Prioridad	Alta
Necesidad	Esencial
Descripción	El videojuego debe ser visualmente intuitivo y aceptable por el usuario.

TABLA 2.32. REQUISITO NO FUNCIONAL – MINIMALISTA

Identificador	**RNF10**
Nombre	Minimalista
Prioridad	Alta
Necesidad	Esencial
Descripción	La interfaz del usuario no deberá estar sobrecargada de elementos.

2.3.4.7. Requisitos no funcionales de datos

TABLA 2.33. REQUISITO NO FUNCIONAL – LENGUAJE DE PROGRAMACIÓN

Identificador	**RNF11**
Nombre	Lenguaje de programación
Prioridad	Alta
Necesidad	Esencial
Descripción	El lenguaje de programación utilizado deberá ser CSharp.

TABLA 2.34. REQUISITO NO FUNCIONAL – FORMATO DE LA MÚSICA

Identificador	**RNF12**
Nombre	Formato de la música
Prioridad	Alta
Necesidad	Esencial
Descripción	El formato de la música deberá ser “.mp3” o “.waw”.

TABLA 2.35. REQUISITO NO FUNCIONAL – FORMATO DE LAS TEXTURAS

Identificador	**RNF13**
Nombre	Formato de las texturas
Prioridad	Alta
Necesidad	Esencial
Descripción	El formato de las texturas deberá ser “.png” o “.jpg”.

TABLA 2.36. REQUISITO NO FUNCIONAL – VERSIÓN DE UNITY

Identificador	**RNF14**
Nombre	Versión de Unity
Prioridad	Alta
Necesidad	Esencial
Descripción	El proyecto se desarrolla a partir de las versiones de Unity 2019.3 y superiores.

2.3.5. Conclusiones

Para obtener un preciso funcionamiento del videojuego, se ha visto la necesidad del obligado cumplimiento de todos los requisitos exceptuando los requisitos del sonido, música e idioma. Esto es porque el principal objetivo del videojuego es la simulación de un ejercicio de rehabilitación, en el cual, el jugador no necesita de música para realizar el ejercicio. Por tanto, la música opcionalmente actuará como una estimulación o relajación del paciente (si se considerase necesario). No obstante, el incumplimiento de cualquiera del resto de los requisitos afectará negativamente a la simulación del proyecto y, por consiguiente, impedirá alcanzar el objetivo deseado.

3. DISEÑO DE LA SOLUCIÓN TÉCNICA

3.1. Introducción

El propósito de esta sección es mostrar el progreso desde una idea inicial del videojuego hasta su implementación completa. Para ello, se barajarán diferentes alternativas de diseño y se seleccionará la alternativa considerada como la más adecuada. También se representarán las técnicas utilizadas para cada programa requerido hasta conseguir el modelo resultante.

3.2. Diseño de la solución inicial

Este subapartado trata de mostrar las ideas principales que se habían desarrollado durante la etapa de diseño inicial previa a la primera programación del software.

3.2.1. Ideas básicas: ¿Cómo será el videojuego?

El objetivo principal es simular un ejercicio donde el jugador deberá tocar obstáculos con forma de esfera lo más próximo al centro posible. Partiendo de dicho objetivo, se alcanzaron las siguientes ideas iniciales:

- El jugador se encontrará en el punto principal del escenario, es decir, se encontrará dentro del campo donde se reproducirán los obstáculos y, por tanto, no deberá desplazarse hacia un punto determinado de encuentro.
- Los obstáculos deberán aparecer estáticos a una distancia alcanzable por los brazos del jugador.
- Los obstáculos deben aparecer en cualquier ángulo del jugador, es decir, pueden aparecer a poca altura para que tenga que agacharse o incluso detrás para que tenga que darse la vuelta.
- Cuando un obstáculo es destruido entonces aparecerá otro obstáculo.
- Existirá un cartel que muestra el tiempo restante de la prueba para que el jugador pueda controlar continuamente cuánto tiempo le queda.
- El escenario será creado por elementos sencillos como poliedros.

3.2.2. Elementos que puede contener un nivel

Se van a distinguir entre distintos elementos que puede contener un nivel:

- Obstáculos de esfera: en función del progreso que vaya alcanzando el jugador, cada partida tendrá una cantidad diferente de obstáculos. A mayor velocidad de reacción ante los obstáculos, el jugador podrá interaccionar con mayor cantidad de ellos.
- Manos: son el principal componente del jugador, ya que, es el elemento que se utiliza para interactuar con los obstáculos.
- Suelo: será la base con una o varias texturas que permitirá el montaje de todo el escenario.
- Posición del jugador: son las coordenadas iniciales donde aparecerá el jugador dentro del escenario.
- Posición de los obstáculos: son las coordenadas posibles de salida que adoptarán los obstáculos al ser creados.

- Parte delantera: se corresponde con la parte del escenario ubicada delante del jugador y que, por tanto, será vista más veces.
- Parte trasera: se corresponde con la parte del escenario ubicada detrás del jugador y que, también será vista por el jugador, pero en menor cantidad que la parte delantera.
- Adornos: son las texturas u objetos utilizados para decorar el escenario.

3.2.3. Software, técnicas y métodos planteados

Entre los objetivos de este proyecto se encuentra el aprendizaje del diseño y modelado de objetos 3D y el aprendizaje de programación y animación de un videojuego.

Diseño y modelado de objetos 3D: trata el modelado de todos los objetos 3D que se muestren en el escenario del videojuego. A pesar de existir librerías y modelos de uso libre y gratuito, el alumno ha decidido diseñar y modelar todos los objetos tridimensionales.

Para el modelado se ha utilizado el programa 3DS MAX que principalmente abarca la generación de gráficos, objetos y animaciones bidimensionales y tridimensionales. Este programa ofrece una licencia gratuita de 3 años a los estudiantes. Además, tiene gran facilidad para instalar plugins gratuitos que incorporan animaciones y modelos ya predefinidos [89]. Se realizará alguna prueba de plugins únicamente con el fin de aprender a utilizarlos, sin embargo, no hará uso de ningún plugin en sus modelos finales para conseguir la mayor instrucción posible sobre este programa.

Los objetos modelados y texturizados con este programa serán exportados y posteriormente importados al programa Unity.

Programación y animación de un videojuego: trata de la creación del escenario del videojuego. El programa principal que será la base del videojuego es Unity, este programa permitirá la creación e importación de los objetos de la escena, inserción de música de fondo, texturización de diferentes elementos, posición de todos los objetos tridimensionales, animación de objetos y programación de funcionamiento de algunos de ellos.

Para alcanzar estos objetivos, se han empleado diferentes técnicas y métodos:

- Scripting: consiste en la creación de ficheros “scripts”. Estos ficheros se utilizan para la implementación del funcionamiento de los objetos con el lenguaje de programación seleccionado. En concreto, en Unity se han creado scripts escritos en lenguaje C Sharp.
- Arrastrar: consiste en arrastrar un ítem a un destino. Se ha utilizado principalmente en Unity al mover objetos entre directorios o al arrastrar objetos a la escena.
- Posicionamiento: consiste en desplazar los objetos tridimensionales por la escena variando así sus coordenadas. Se ha utilizado tanto en Unity como en 3DS MAX.
- Steam VR: consiste en un paquete descargable de libre uso de la “asset store” que permitirá insertar directamente un jugador con el funcionamiento base de realidad virtual: control de los mandos y el visor, reproducción en los visores, detección de altura del jugador y su desplazamiento sobre el entorno real…

3.2.4. Desarrollo de alternativas de diseño

El diseño final seleccionado ha sido el resultado de la decisión entre diferentes alternativas de diseño.

Cada alternativa se ha generado a partir de la alternativa anterior. Por tanto, se entiende que las alternativas evolucionan de manera progresiva y, aquello que no se mencione en una alternativa es porque utiliza lo mismo que la alternativa anterior.

3.2.4.1 Alternativa 1

Partiendo de las ideas iniciales del videojuego, se alcanzaron las primeras percepciones:

- El escenario debería ser los más simple posible para evitar distracciones durante la partida. Por tanto, lo óptimo sería la menor la cantidad posible de decorados y elementos estéticos.
- El obstáculo con forma de esfera debería tener un único diseño para no confundir al jugador ante posibles obstáculos nuevos imprevistos.
- Los obstáculos deberían estar rotando sobre su propio eje a una velocidad mínima para que no provoquen malestar en el jugador y así obtener un efecto de animación visual.
- El modelado de los objetos sería con el programa 3DS MAX.
- El desarrollo del videojuego sería con el programa Unity.

3.2.4.2. Alternativa 2

De la alternativa 1 se observaron algunos inconvenientes con los obstáculos. Por tanto, se intentó cambiar el enfoque con ellos.

- El escenario debería continuar siendo simple. El jugador tiene que tocar los obstáculos y no debe perderse la vista en otros objetos decorativos.
- Deberían existir al menos dos diseños de obstáculos para conseguir escapar de la monotonía que podría crear el videojuego y así, evitar un ciclo repetitivo.
- El rango de las posiciones de la aparición de los obstáculos debería de limitarse. Por tanto, se permite que aparezcan tanto detrás del jugador como delante, no obstante, no deben estar ni muy abajo ni muy encima para impedir que el jugador haga demasiado esfuerzo en agacharse o en estirarse hacia arriba.

3.2.4.3. Alternativa 3

De la alternativa 2 se vio poca utilidad con las manos. Por tanto, se intentó cambiar el enfoque con ellas.

- El escenario debería de tener algún adorno para aumentar la inmersión y orientación del jugador dentro del escenario. Ya que, si no hay adornos, el jugador puede empezar a perder el sentido de la orientación y olvidar qué parte del escenario se corresponde con delante y detrás con respecto a la posición inicial.
- Para tocar los obstáculos no se distinguía entre ninguna de las dos manos. Esto lleva a que el jugador puede entrar en una rutina de utilizar siempre una mano y no entrenar la otra. Por consiguiente, cada mano podrá interactuar con un obstáculo correspondiente.

- Para distinguir a qué mano pertenece cada obstáculo, se utilizará un color diferente. Lo que implica que los obstáculos deberán de tener dos diseños y cada una deberá ser de un color igual a su respectiva mano.
- Entonces se genera la posibilidad de que el jugador golpee un obstáculo con la mano equivocada, ese caso se considerará como precisión de 0.

3.2.4.4. Alternativa 4

Con la alternativa 3 se consigue un videojuego funcional, pero genera demasiadas incomodidades al jugador.

- El escenario debería mostrar un lugar que sea cómodo para el jugador. Este lugar debería ser más complejo con objetos tridimensionales decorativos que hagan notar que su utilidad es únicamente estética y no forman parte de la funcionalidad del juego.
- Teniendo en cuenta que las posiciones de aparición de los obstáculos varían tanto delante como detrás del jugador, es posible que el jugador se maree y no sea capaz de terminar la prueba. Por tanto, las posiciones de aparición de los obstáculos estarán ubicadas en la parte delantera.

3.2.4.5. Alternativa 5

Con la alternativa 4 existen problemas imprevistos por parte del jugador que serían difíciles de controlar.

- Si son demasiado cercanas las posiciones de aparición de los obstáculos al jugador, es posible que toque los obstáculos de manera no intencionada y, por tanto, los resultados obtenidos no estarían midiéndose correctamente. Para conseguir que el obstáculo se toque con un tiempo de respuesta suficiente para el usuario, se colocarán los obstáculos a una distancia lejana. Esto implica que los obstáculos deberán de acercarse al jugador a una velocidad determinada.
- Para evitar que la prueba se haga larga y tediosa, existirá un tiempo para la próxima aparición del obstáculo.
- Debido a que los obstáculos aparecerán en una tanda seguida, el jugador no podrá estar moviéndose rápidamente de posición. Por lo que, la posición de aparición de los obstáculos se reduce a dos valores posibles. Una posición se correspondería con la mano derecha y la otra con la mano izquierda.
- El movimiento de los obstáculos deberá tener una dirección paralela al suelo y un sentido que lo aproxime a la posición inicial del jugador.
- Puesto que los obstáculos se aproximan al jugador, existirá la posibilidad de que el jugador no la intercepte (si no llegase), por lo que, los obstáculos podrán atravesar al jugador y continuar su dirección.
- Al finalizar el tiempo de duración de la prueba, es posible que haya obstáculos que estuvieran delante del jugador. Esos obstáculos no formarán parte en el recuento de los resultados.
- Como el juego aumenta su complejidad, se podrán obtener todavía más resultados al finalizar la partida: duración de la partida, la velocidad de movimiento de los obstáculos, el intervalo de tiempo de salida entre cada obstáculo, la cantidad total de obstáculos de cada color y totales, la cantidad de obstáculos golpeados de cada color y totales, la cantidad de obstáculos no

golpeados de cada color y totales, la cantidad de obstáculos golpeados incorrectamente de cada color y totales y el total de precisión del jugador con los obstáculos de cada color y totales.

3.2.4.6. Alternativa 6

La alternativa 5 resuelve los problemas planteados anteriormente, sin embargo, recupera la monotonía.

- Si los obstáculos aparecen en dos posiciones posibles y siempre el obstáculo derecho corresponde con la mano derecha y el obstáculo izquierda con la mano izquierda, el jugador rápidamente se aburrirá en la misma mecánica. Por tanto, el obstáculo que se genera en una de las dos posiciones será aleatoria, es decir, es posible que en la posición derecha aparezca un obstáculo que se debe tocar con la mano izquierda y viceversa.

3.2.5. Resumen y elección del diseño

A continuación, se va a mostrar una tabla ilustrativa con las características principales de cada alternativa.

- Escenario: nivel de complejidad que tiene el escenario. Existen 3 niveles: simple, medio, complejo.
- Monotonía: indica si el jugador alcanzará rápidamente aburrimiento por ciclo repetitivo. Existen 2 posibilidades: Sí, No.
- Orientación: representa si el jugador tendrá facilidad para perder el sentido de la orientación. Existen 2 posibilidades: Sí, No.
- Color de obstáculos: indica la cantidad de colores que tiene el obstáculo. Existen 2 posibilidades: unicolor de un color y bicolor de dos colores.
- Color de manos: refleja la cantidad de colores que tienen las manos. Existen 2 posibilidades: unicolor de un color y bicolor de dos colores.
- Posición obstáculos: muestra la posición de aparición de los obstáculos en relación con la posición inicial del jugador.
- Pulsación inesperada: indica si el jugador tocará un obstáculo de forma involuntaria. Existen 2 posibilidades: Sí, No.

TABLA 3.1. COMPARATIVA DE LAS ALTERNATIVAS DE DISEÑO

Alternativa	Escenario	Monotonía	Orientación	Color de obstáculos	Color de manos	Posición obstáculos	Pulsación inesperada
1	Simple	No	Sí	Unicolor	Unicolor	Delante y atrás	Sí
2	Simple	No	Sí	Unicolor	Unicolor	Delante y atrás	Sí
3	Medio	No	No	Bicolor	Bicolor	Delante y atrás	Sí
4	Complejo	No	No	Bicolor	Bicolor	Delante	Sí
5	Complejo	Sí	No	Bicolor	Bicolor	Delante	No
6	Complejo	No	No	Bicolor	Bicolor	Delante	No

Lo más importante para seleccionar un diseño es buscar la comodidad del jugador, por tanto, las dos primeras alternativas se descartan directamente porque es crucial que el jugador no pierda el sentido de la orientación.

El jugador está realizando un ejercicio y se busca evaluar sus resultados con la mayor precisión posible. Si el jugador entra en una rutina o en una monotonía, es probable que deje de concentrarse y empiece a ejecutar peor el ejercicio, por consiguiente, la alternativa 5 se desecha. Además, debe interaccionar con los obstáculos de manera voluntaria para medir su toque con tiempo, con la alternativa 4 puede golpear obstáculos antes de tiempo y empeorar sus resultados, así pues, se rechaza esta alternativa.

El jugador no debe estar buscando continuamente el obstáculo, si no principalmente debe tener buena precisión a los obstáculos que se les plantea. Es por ello, por lo que la alternativa 3 se excluye, ya que, implicaría al jugador estar continuamente dando vueltas buscando obstáculos.

Finalmente, la alternativa que mantiene más comodidad al jugador y permite obtener los resultados con la mayor precisión posible es la alternativa 6. Por todo ello, será seleccionada dicha alternativa.

3.3. Desarrollo del diseño final

3.3.1. Introducción

El propósito de esta sección es mostrar un crecimiento progresivo de la implementación de la alternativa de diseño seleccionada. Primero se va a exponer el funcionamiento principal del videojuego. Posteriormente, se explicará el modelado del escenario.

El modelado del escenario comenzará con un escenario simple. El modelado se presentará siguiendo una secuencia explicativa de pasos: modelado y texturización del entorno, modelado de los objetos funcionales (obstáculos y manos) y texturización de los mismos, modelado de elementos decorativos y uso de luces.

Con el escenario simple se obtendrá mayor visión espacial para permitir la creación del escenario complejo. El modelado del escenario complejo seguirá un desarrollo detallado secuencial: modelado y texturización del entorno, modelado, texturización y animación de los objetos funcionales (obstáculos y manos) y modelado y texturización de los elementos decorativos.

3.3.2. Funcionamiento principal

El videojuego va a mostrar al usuario una secuencia de obstáculos de un determinado color que se irán acercando al usuario y deberá tocarlas con la mano correspondiente lo más próximo al centro posible.

Para variar la dificultad en función del progreso que vaya alcanzando el jugador, existirán una serie de parámetros que se podrán modificar: posición del obstáculo, velocidad de los obstáculos, intervalo de salida entre cada obstáculo y duración de la prueba.

Para la evaluación del ejercicio se tendrá en cuenta la precisión que ha tenido el jugador para todos los obstáculos y para cada color, la cantidad de obstáculos que no

ha alcanzado totales y de cada color y la cantidad de obstáculos que ha golpeado con la mano equivocada totales y de cada color.

3.3.3. Diseño, modelado y animación de objetos tridimensionales

La adquisición del conocimiento sobre las diferentes funcionalidades, herramientas y opciones que ofrece el programa Unity y Autodesk 3DS MAX, se ha obtenido por sus respectivos manuales [90] y [91] y páginas de foros y videotutoriales entre otras fuentes.

3.3.3.1. Escenario simple

El propósito del escenario simple es crear una primera versión del escenario del videojuego para crear una imagen principal sobre el diseño inicial.

Entorno

Corresponde con el ambiente donde se realizará la prueba del usuario. Puesto que los obstáculos tendrán que venir por el frente, lo primero que se ha pensado es en realizar un pasillo para dar pie al recorrido que tendrían que realizar los obstáculos hasta llegar al usuario.

Para ello, mediante el programa 3DS MAX se posicionarán 5 cajas para que cada una actúe de una determinada forma: un suelo, dos paredes, un techo y un fondo.

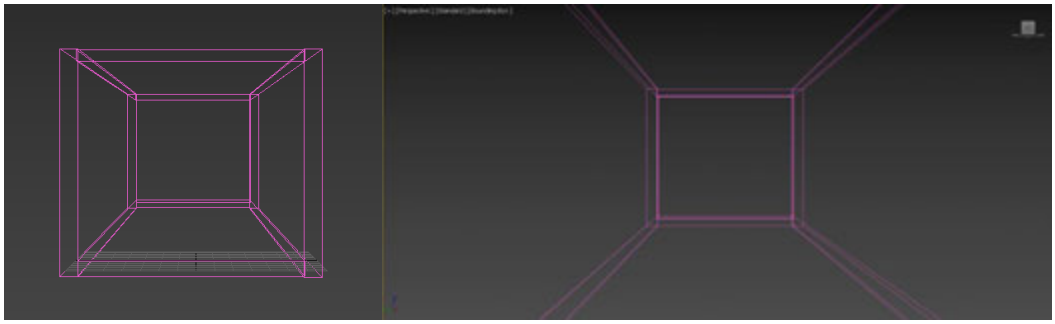


Fig.22 – Diseño del pasillo delantero al usuario en 3DS MAX.

Una vez diseñado el pasillo, hay que probarlo con cajas reales.

Fig.23 – Pasillo delantero al usuario en 3DS MAX.

De esta forma, ya se podría establecer la posición inicial del jugador, siendo el centro del pasillo.

A pesar de que la funcionalidad del videojuego se centrará en la parte delantera del usuario, también hay que tener en cuenta que el usuario puede darse la vuelta (si lo desea). Por tanto, también habrá que continuar el pasillo para que el usuario no se encuentre con un espacio vacío detrás de sí mismo.

Fig.24 – Diseño del pasillo trasero al usuario en 3DS MAX.

No es necesario ocultar el final del pasillo, ya que, el usuario nunca llegará a pensar que ese final se trata del espacio vacío.

Con cajas reales, el final del pasillo sería el siguiente:

Fig.25 – Pasero trasero al usuario en 3DS MAX.

Se puede observar cómo al final se pierde la noción de ser el espacio vacío.

Finalmente, hay que aplicar una textura al pasillo resultante. Como el usuario estará constantemente observando involuntariamente el pasillo, es importante no seleccionar colores claros para evitar reflejos y dificultades para encontrar los obstáculos. Por tanto, se ha decidido seleccionar un color negro con tonalidad oscura.

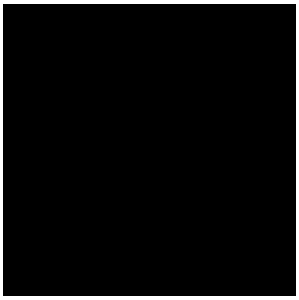


Fig.26 – Textura utilizada para el pasillo.

Una vez exportado a Unity, con las propiedades del reflejo de colores “reflection”, se podrá conseguir que esta textura refleje la luz para así conseguir un pasillo oscuro, pero sin perder alta cantidad de iluminación.

Fig.27 – Pasillo delantero y trasero resultante en Unity.

De esta forma, el usuario puede ver el entorno con claridad.

Objeto funcional: obstáculo

El diseño inicial de obstáculo por el que se ha optado es el de una esfera en sí misma, es decir, sin deformidades.

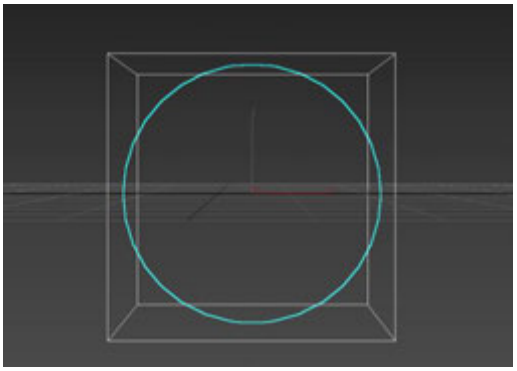


Fig.28 – Diseño de obstáculo simple en 3DS MAX.

Una vez diseñado el obstáculo, se prueba con esferas reales.

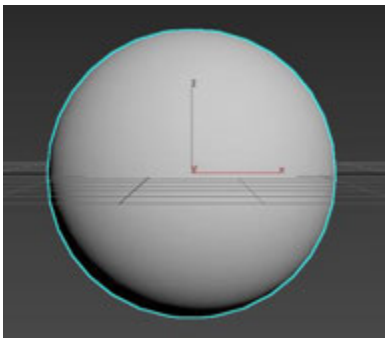


Fig.29 – Obstáculo simple en 3DS MAX.

El siguiente paso es aplicar una textura sobre los obstáculos. El propósito es entender claramente la diferencia entre un obstáculo y otro para así saber a qué mano corresponderá cada uno, por tanto, el color de cada uno debe ser muy diferente. Es por ello, por lo que se ha decidido utilizar el color rojo y azul.

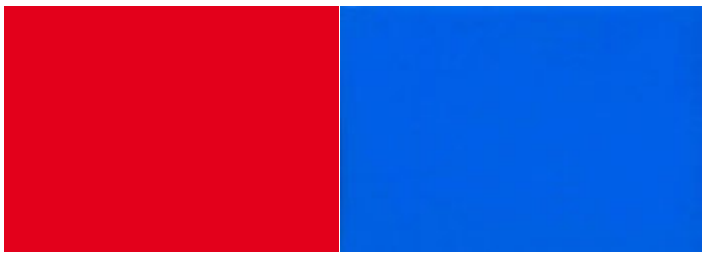


Fig.30 – Texturas de obstáculo simple: rojo y azul.

Una vez exportado a Unity, se utilizarán las propiedades del reflejo del color para conseguir un mejor acabado, se texturizará el centro de las esferas con mayor claridad y se posicionarán los obstáculos al final del pasillo.

Fig.31 – Obstáculos simples lejos y cerca del usuario en Unity.

De esta forma, ya se tendrían los obstáculos diferenciados y su posición de aparición.

Objeto funcional: manos

Las manos serán el arma que utilizará el usuario para eliminar los obstáculos, por tanto, realizar unas manos realistas como armas podría resultar extraño al usuario.

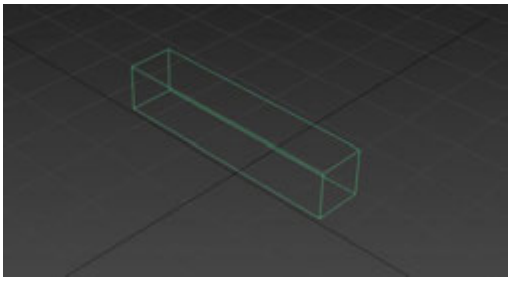


Fig.32 – Diseño de mano simple en 3DS MAX.

Se ha decidido utilizar un prisma rectangular como arma, puesto que puede simular el comportamiento de una espada o un palo que será capaz de eliminar a los obstáculos.

A continuación, se procede con la creación del prisma.

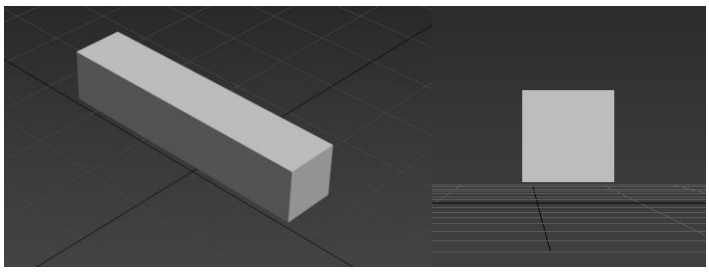


Fig.33 – Mano simple en 3DS MAX.

Las texturas que se utilizarán para la mano deben corresponderse con las mismas del obstáculo, así será más sencillo distinguir a qué mano pertenece cada obstáculo. En Unity se obtendría el siguiente resultado:

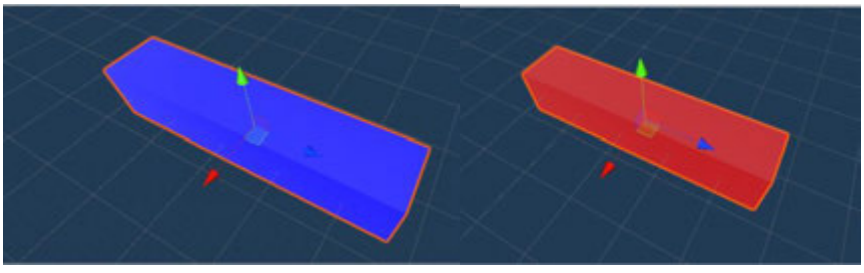


Fig.34 – Mano simple texturizada en Unity.

Se aprecia correctamente la mano correspondiente a cada obstáculo.

Decoración del entorno

Los elementos que se utilizan en la decoración deben seguir el mismo estilo que los elementos que componen el entorno. Para este entorno, el estilo de los elementos de decoración deben ser poliedros.

El entorno consiste en un pasillo. Un pasillo cerrado está poco iluminado y la oscuridad puede generar sentimientos negativos por parte del jugador. Es por ello, por lo que se ve la necesidad de crear luces que oculten la oscuridad.

El diseño que van a adoptar para las luces serán prismas rectangulares estrechos.

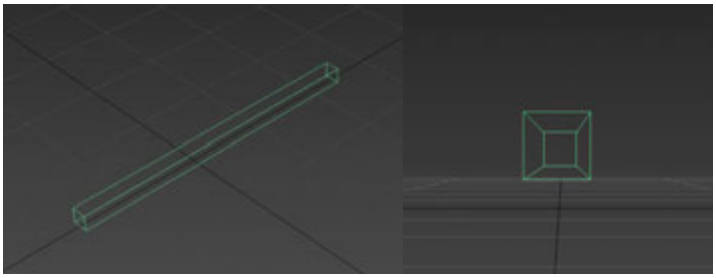


Fig.35 – Diseño de la barra luminosa en 3DS MAX.

Si se crea esta barra, se obtiene lo siguiente:

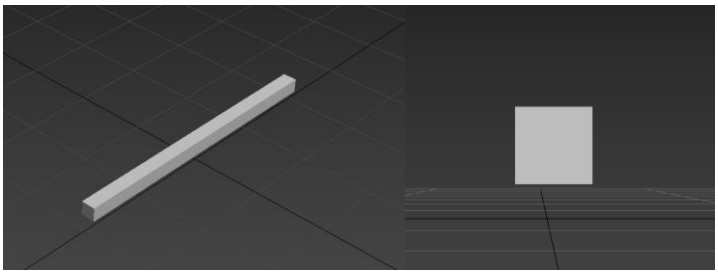


Fig.36 – Barra luminosa en 3DS MAX.

El color principal que debe adoptar esta barra de luz debe ser un color claro. Sin embargo, hay que tener cuidado con esta elección, ya que, hay colores que podrían dificultar la visualización de los obstáculos. Por ejemplo, luces de color azul y rojo podrían hacer confundir sobre las dimensiones del obstáculo. Los colores más recomendables serían blanco o amarillo.

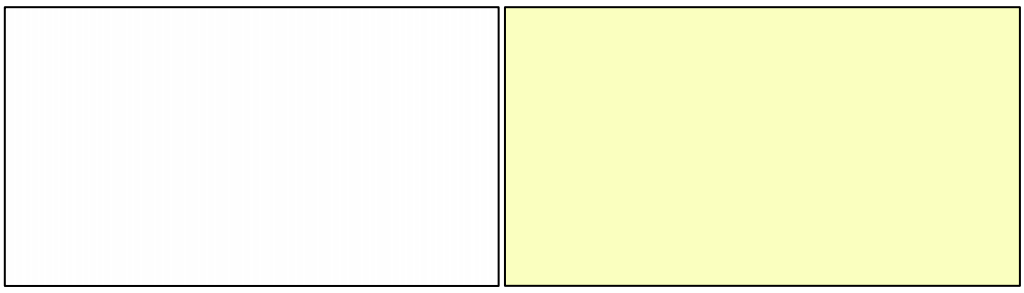


Fig.37 – Posibles texturas de la barra luminosa: blanco y amarillo claro.

Mantener el mismo estilo de colores en el entorno es un factor muy importante, por tanto, el color blanco puede actuar como contraste frente al color oscuro lo que

crearía un estilo artístico de negativos. Es decir, un estilo donde se aplican los colores opuestos entre sí. Por lo que, seleccionar el color blanco sería lo más adecuado.

Al exportar la barra a Unity, se obtendría el siguiente resultado:

Fig.38 – Barra luminosa en Unity.

La barra se ha ubicado en el centro del suelo para que el usuario siempre tenga percepción del centro del suelo del pasillo.

Se puede observar que, debido a la oscuridad del entorno, el blanco de la barra se pierde y termina en un color de gris claro. Para arreglar este problema, dentro de las propiedades del color de Unity, existe la propiedad “Emission”, esta propiedad permite que un color genere una luz por sí mismo de un color determinado, en este caso, se ha elegido el blanco.

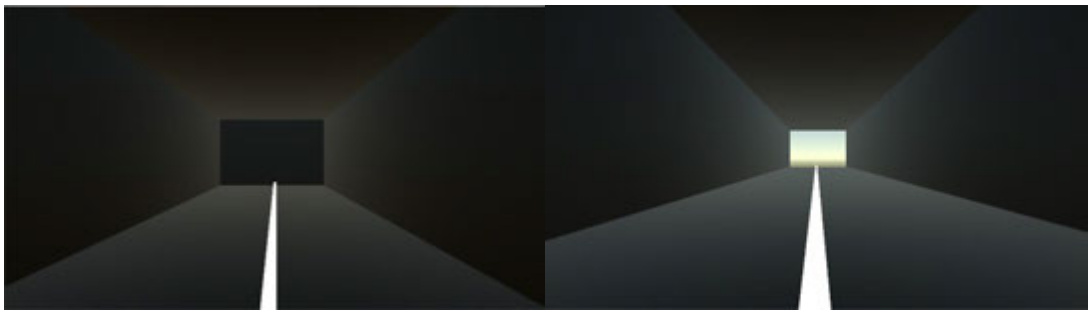


Fig.39 – Barra luminosa con emisión de luz blanca en Unity.

Se puede observar mejoría en el tono de claridad que alcanza el blanco y en la iluminación que genera. Además, la intensidad de la luz no es lo suficientemente alta como para generar reflejos que dificulten la visualización del entorno.

Aprovechando esta propiedad y el efecto que genera, se van a añadir más luces en el entorno para mejorar la estética del pasillo.

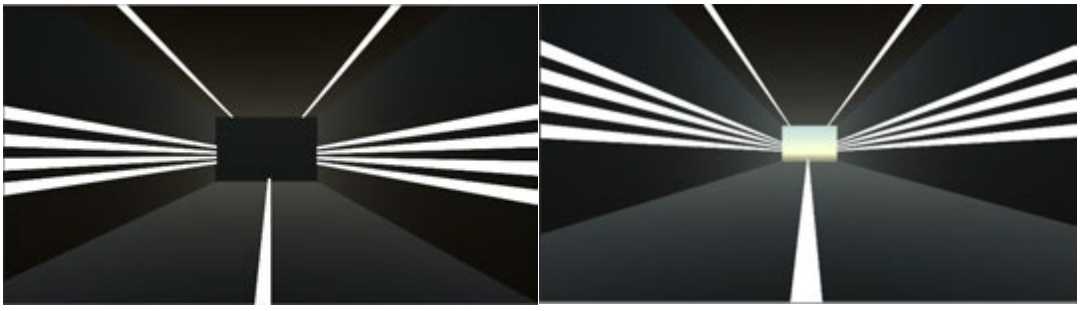


Fig.40 – Incorporación de todas las barras luminosas con emisión de luz blanca en Unity.

Se obtiene un pasillo correctamente iluminado y sin ningún tipo de reflejos.

Conclusiones del escenario simple

Finalmente, el resultado del escenario simple es el siguiente:

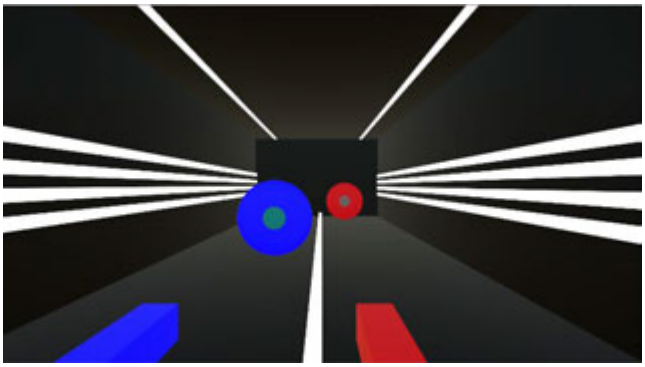


Fig.41 – Resultado del escenario simple.

Para el proyecto, este escenario es muy útil para obtener las dimensiones de los obstáculos y su posición inicial de aparición y la distancia existente entre la posición inicial del jugador y la posición de aparición de los obstáculos. También, ha servido como una introducción al manejo de cada uno de los programas utilizados.

Para el jugador, se podría utilizar este escenario como un nivel inicial para que caliente y obtenga una idea de lo que será su prueba.

3.3.3.2. Escenario complejo

El propósito de este escenario es crear la base definitiva sobre la que se desarrollará el videojuego.

Para crear el escenario primero hay que definir la temática que se pretende seguir. Existen diferentes temas que se podrían llevar a este videojuego: campo de fútbol donde el jugador es el portero y tiene que parar balones, campo de gladiadores donde el jugador porta una espada y tiene que romper cascos de enemigos, una galaxia donde el jugador porta espadas laser y tiene que destruir marcianos…

Fig.42 – Temáticas posibles: campo de fútbol [92], campo de gladiadores [93] y galaxia [94].

Hay que tener en cuenta que el jugador es un paciente que ha sufrido un trastorno neurológico, por tanto, el entorno debería representar un ambiente tranquilo y amigable para el jugador.

Los escenarios que visualmente son atractivos y donde un jugador se siente cómodo son las praderas, bosques, lagos… Entornos con un paisaje agradable que inciten a la serenidad del jugador y así favorecer su concentración.

Realizar un entorno con un paisaje sereno no implica necesariamente un desagrado hacia el jugador. Existen multitud de videojuegos que han alcanzado

millones de ventas y muestran paisajes de sosiego y reposo. Juegos como “The Elder Scrolls V: Skyrim”, “World of Warcraft”, “The Witcher 2: Assassins of Kings”, “Red Dead Redemption”, “Uncharted 2: Among Thieves”, “Myst IV: Revelations”, “BioShock Infinite” … [107]. De esta forma, se ha decidido utilizar un entorno que representa un bosque.

Para completar la temática del videojuego, hay que definir cuál será la motivación para que el usuario quiera que destruir los obstáculos con el mayor gusto posible. De un bosque se obtienen diferentes ideas: el jugador debe disparar dianas con flechas, el jugador debe talar troncos de árbol con hachas, el jugador debe plantar árboles en el bosque… Este tipo de tareas son las que se realizan en un bosque, sin embargo, se alejan del objetivo principal del videojuego: tocar obstáculos lo más al centro posible con las manos. Por tanto, trasladar estas tareas en forma de “tocar un objeto” perdería gran parte de su sentido.

Debido a estas dificultades de traslación de tareas al videojuego, se barajará otras opciones. En vez de acercarse a una representación realista, se podría acercar a una representación de fantasía.

La asociación de los bosques con un mundo mágico se lleva realizando durante más de 30 años. Esta asociación se ha llevado a todo tipo de contenido multimedia.

Se puede destacar películas sobre magia populares como “Stardust”, “Harry Potter”, “Black Forest”, “Cudesna suma”, “Wizard of Oz” o “Willow”. Todas estas películas que utilizan el concepto de magia contienen escenas de bosques misteriosos con magia oculta en su interior [108].

Los videojuegos con temática de magia también utilizan los bosques. Se puede encontrar juegos como “Outward”, “Spellbreak”, “Tyranny”, “Magicka”, “In Verbis Virtus” donde explotan estos escenarios [109].

Se ha podido observar que, si se utiliza una temática de magia sobre el entorno del bosque, no será algo imprevisto para el jugador porque se ha utilizado durante muchos años y en todo tipo de contenido audiovisual.

En conclusión, el juego se basará en un entorno de bosque donde el jugador deberá destruir obstáculos misteriosos con el contacto de varitas mágicas.

Entorno

Existe multitud de tipos de bosques. Según un criterio propio, se va a realizar una clasificación con los principales tipos de bosques:

- Bosques poblados: son bosques repletos de árboles donde se dificulta bastante la visualización del fondo del escenario, ya que, se esconde por todos los árboles. Suelen ser árboles muy altos que necesitan de mirar hacia arriba para ver sus copas del árbol.
- Bosques dinámicos: son bosques que contienen menor cantidad de árboles y con menor tamaño que los bosques poblados. Es fundamental que aparezca un río donde adquiere gran protagonismo las corrientes del agua del río.
- Bosques comarcales: son un híbrido entre bosques y praderas. Son bosques con menor cantidad de árboles y del tamaño suficiente para ver su copa sin necesidad de erguir la cabeza hacia arriba, pero, también contienen abundante césped a modo de pradera.

- Bosques apaciguados: son bosques donde el protagonismo lo adquiere un lago. El lago principalmente se ubica en el centro de la escena y los árboles bordean el lago embelleciendo la escena.
- Bosques acantilados: están divididos en dos partes. La primera parte es la más cercana al usuario donde se muestra un terreno con una caída pronunciada a modo de acantilado o precipicio. La segunda parte se corresponde con el fondo de la escena que suele ser un bosque poblado para dar a entender que la caída del precipicio te llevaría al bosque.

Fig.43 – Clasificación de bosques: bosques poblados [95], bosques dinámicos [96] y bosques comarcales [97].

Fig.44 – Clasificación de bosques: bosques apaciguados [98] y bosques acantilados [99].

Hay que tener en cuenta que, el usuario debe sentirse cómodo dentro del escenario y no debe perder la atención en el fondo para evitar afectar negativamente en sus resultados de la prueba.

La aparición de un río con fuerte corriente es un motivo principal para la desconcentración del jugador. Por otra parte, si se utiliza un lago, el usuario quedaría flotando sobre el mar o sobre un puente, pero en ambos casos se generaría una sensación de miedo a no caer al mar durante la partida. Por tanto, no se utilizará un bosque dinámico ni un bosque apaciguado.

Si el usuario se expone ante un precipicio o una rampa muy pronunciada, es posible que genere una sensación de vértigo y no realice los ejercicios correctamente. Dicho esto, el bosque acantilado debe descartarse.

Si el bosque está muy poblado de árboles, será muy fácil perder de vista a los obstáculos y, por tanto, el usuario perdería gran parte de su precisión. Entonces, los bosques poblados también se eliminan como opción.

Finalmente, el bosque más adecuado para esta prueba sería un bosque comarcal, ya que, contiene la cantidad suficiente de árboles y otros decorados para embellecer la escena, pero, sin obstaculizar el rumbo de los obstáculos.

Terreno

Unity es capaz de crear planos para que actúen como un terreno.

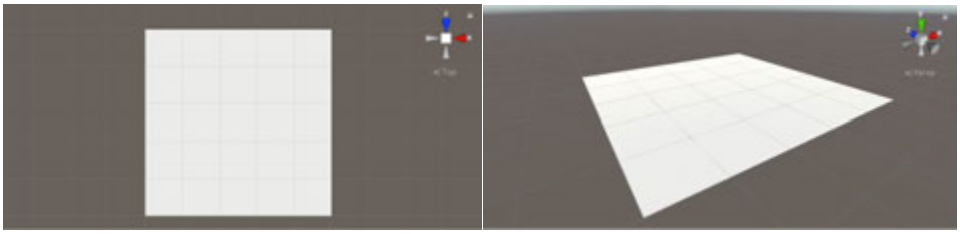


Fig.45 – Terreno inicial en Unity.

Lo primero que se hace es colocar la posición del inicio del jugador en el centro del plano para perfeccionar aquel terreno que se encuentre dentro de su campo visual. Puesto que inicialmente el plano es llano, el jugador vería totalmente el terreno.

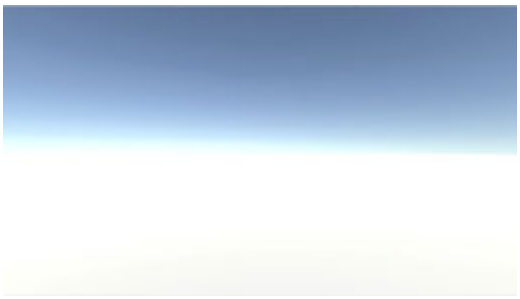


Fig.46 – Vista inicial del jugador en el terreno inicial.

Si el terreno es completamente blanco y se aplican deformidades, resulta más difícil de apreciar los cambios que se van aplicando. Por tanto, se va a texturizar el suelo con hierbas para que adquiera visión de terreno real y sea más fácilmente visible las posteriores deformidades.

Fig.47 – Textura del suelo del terreno [100].

Con esta textura elegida, el suelo apenas difumina.

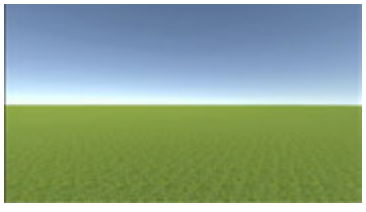


Fig.48 – Vista inicial del jugador en el terreno inicial texturizado.

En todos los terrenos de bosques que se han contemplado, no existe un bosque compuesto por un terreno completamente llano. Unity contiene la herramienta “Raise or Lower Terrain” que permite ensanchar zonas de espacio limitado hasta una altura

concreta. De esta forma, se podrá crear ondulaciones en el terreno para definir el camino sobre el que circularán los obstáculos.

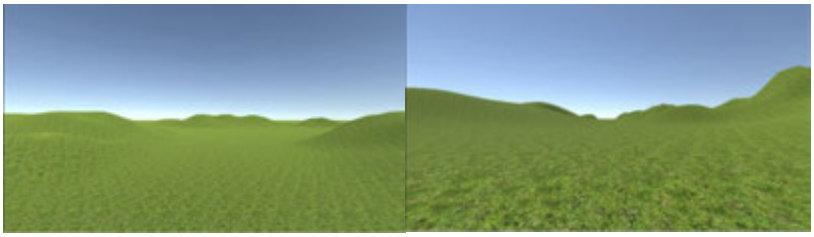


Fig.49 – Vista inicial delantera y trasera del jugador del camino de circulación de obstáculos.

Al crear pequeños montículos sobre el terreno, se crea mayor realidad sobre el mismo. Además, estos montículos limitan el campo visual del jugador, por lo que, se pierde la percepción de espacio vacío del fondo.

Hay que destacar que no se han aplicado deformidades sobre todo el terreno, si no sobre una parte pequeña del mismo. Pero al combinar el tamaño y la distancia de cada montículo con el jugador, no se necesita de crear más deformidades sobre el terreno restante porque estaría siendo tapados por los montículos más cercanos al jugador.

Fig.50 – Camino de circulación de obstáculos desde arriba.

Al haber creado los montículos del terreno, se puede diferenciar el campo o área del entorno que será trabajado y el que no será trabajado porque nunca será visto por el jugador.

Para dar la sensación de que el usuario se encuentra en un bosque dentro de la naturaleza, debería de existir al menos una montaña en el fondo. Para ello, se aplicará la misma herramienta de los montículos.

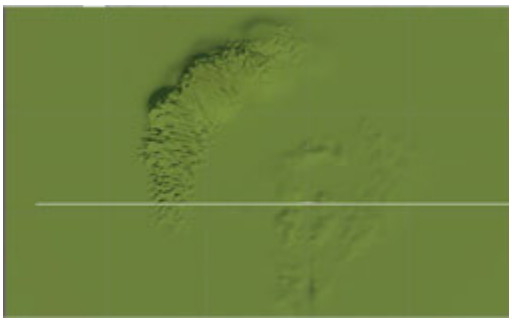


Fig.51 – Modelado del terreno resultante visto desde arriba.

La montaña se ha creado mediante la generación de montículos muy próximos entre sí para aumentar su altura total y conseguir deformaciones entre ella, así dan a parecer que son piedras.

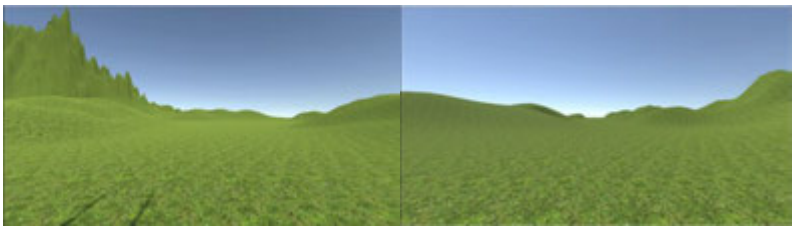


Fig.52 – Vista inicial delantera y trasera del jugador del modelado del terreno resultante.

El jugador puede observar la montaña con un aspecto de composición de rocas puntiagudas.

Texturizado del terreno

Las texturas que se aplican al entorno deben ser lo más claras posibles para ayudar la representación de la escena.

En primer lugar, se va a texturizar el recorrido que realizarán los obstáculos. De esta forma, el usuario sabrá dónde tiene que mantener la mirada fija. Puesto que el escenario está simulando un bosque cerca de montañas rocosas, el camino debería ser de tierra molida.

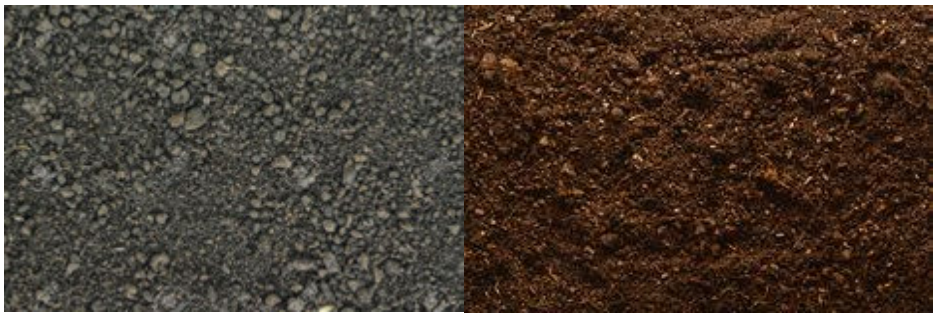


Fig.53 – Texturas del recorrido de los obstáculos [101].

Se va a utilizar una mezcla de las dos texturas de tierra mostradas. Primero se utilizará la tierra oscura porque generalmente el terreno a mayor profundidad es más oscuro y encima se situará la primera textura.

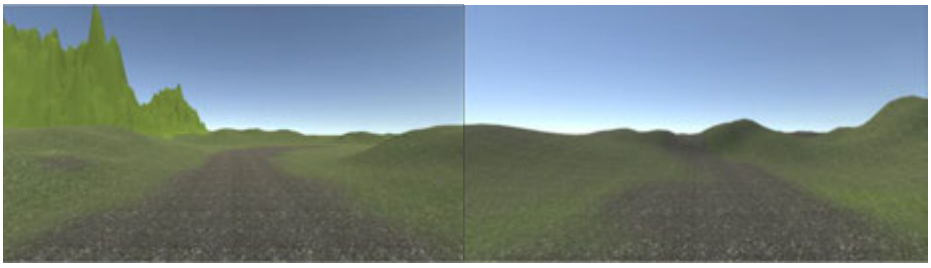


Fig.54 – Vista inicial delantera y trasera del jugador del trazado de los obstáculos.

El jugador puede observar el trazado que seguirán los obstáculos.

Pata terminar el terreno hay que texturizar la montaña del fondo. Como se pretende mantener el mismo estilo de colores en toda la escena, se ha decidido utilizar la misma capa externa de textura que se aplicó para el suelo. Además, al tratarse de un elemento del fondo, resulta más complicado percibir que es la misma textura.

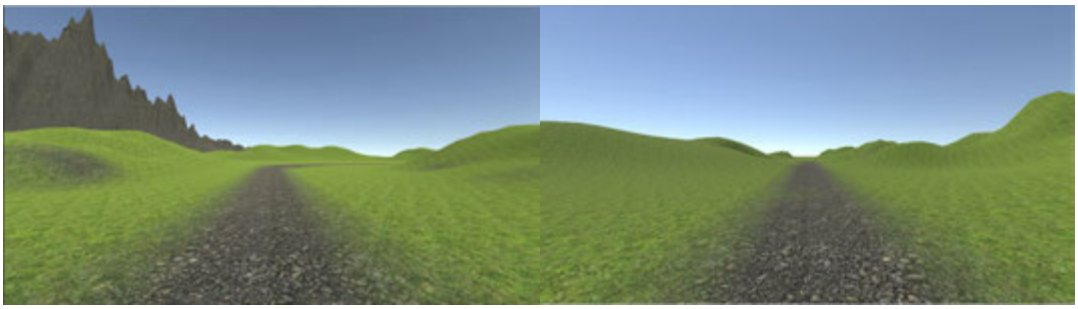


Fig.55 – Vista inicial delantera y trasera del jugador del entorno resultante.

Objeto funcional: obstáculos

Para incorporar la temática de la magia a la escena, se deberá de crear un modelado de obstáculos capaz de permitir la facilidad visual de las animaciones que tendrá integradas.

A fin de facilitar la visualización de las animaciones, se cambiará el diseño del obstáculo de esfera opaca por una tubería.

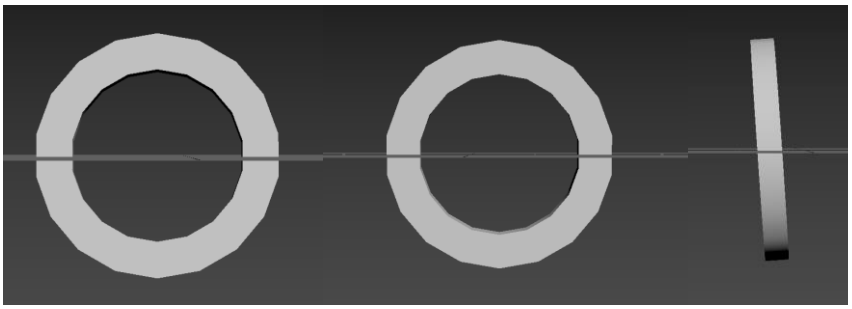


Fig.56 – Modelado del obstáculo esfera inicial: frente, trasero y lateral izquierdo.

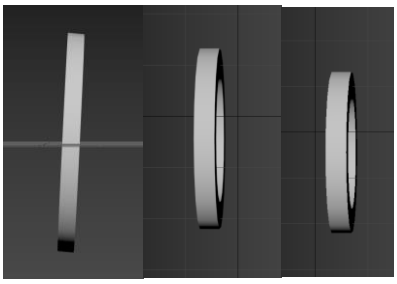


Fig.57 – Modelado del obstáculo esfera inicial: lateral derecho, arriba, abajo.

No obstante, hay que aplicar deformidades sobre la tubería para obtener un acabado más misterioso. Para ello, existe la propiedad “displace” que consiste en la cantidad de desplazamiento de la diagonal imaginaria que compone a la tubería. Si esta diagonal se desplaza afecta a todas las superficies de la tubería, por lo que se consigue deformar al objeto. También existe la variable “strength” que se utiliza para medir la intensidad de deformidad al aplicar un desplazamiento de la diagonal. Por tanto, si el desplazamiento es escaso pero la intensidad elevada la deformidad será mayor y viceversa.

Ajustando estos valores, se ha obtenido el siguiente resultado:

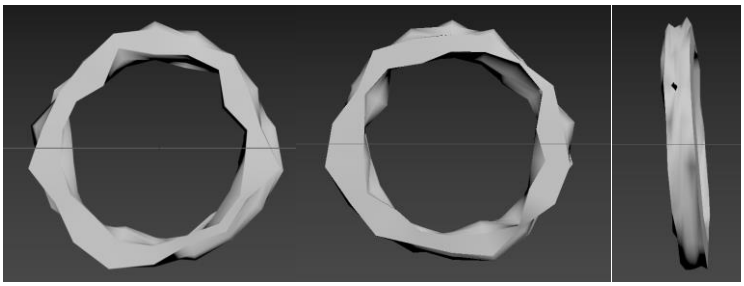


Fig.58 – Vistas del obstáculo esfera sin texturizar: frente, trasero y lateral izquierdo.

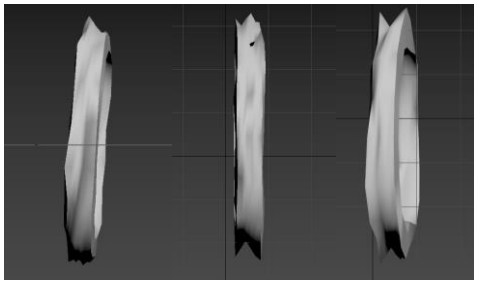


Fig.59 – Vistas del obstáculo esfera sin texturizar: lateral derecho, arriba, abajo.

Texturizado de obstáculos

Ya se explicó anteriormente que los obstáculos serán de color rojo y azul. Para otorgar un aspecto mágico se utilizará una textura de magma para el obstáculo rojo y una textura de agua para el obstáculo azul.

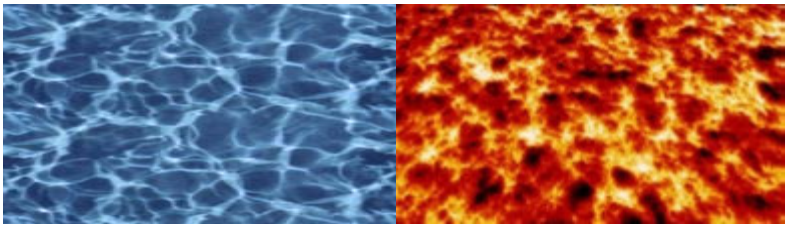


Fig.60 – Textura de los obstáculos [100].

Desde la época de los griegos, ya se utilizaba una diferenciación de cuatro elementos que componen todos los materiales del planeta tierra: agua, fuego, tierra y aire. Con estas texturas se podría asociar a dos de los principales elementos: agua y fuego.

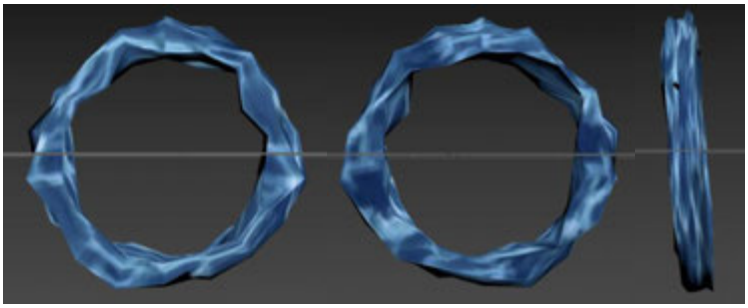


Fig.61 – Vistas del obstáculo esfera azul texturizada: frente, trasero y lateral izquierdo.

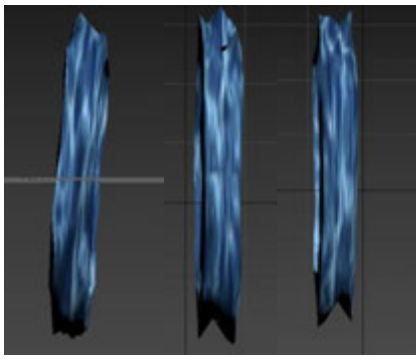


Fig.62 – Vistas del obstáculo esfera azul texturizada: lateral derecho, arriba, abajo.

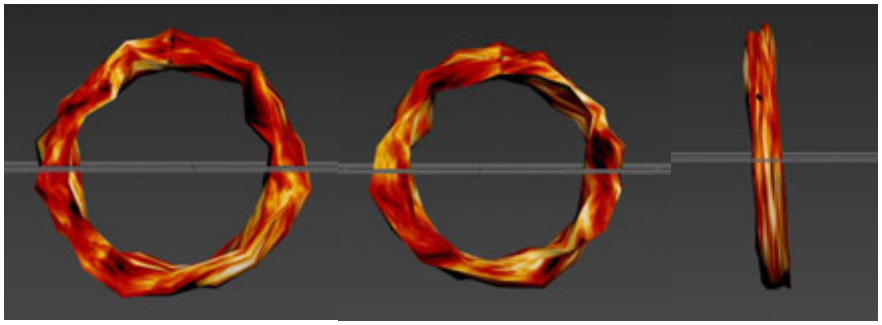


Fig.63 – Vistas del obstáculo esfera roja texturizada: frente, trasero y lateral izquierdo.

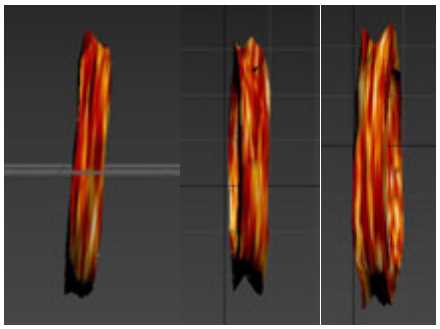


Fig.64 – Vistas del obstáculo esfera roja texturizada: lateral derecho, arriba, abajo.

Animación de los obstáculos

Los obstáculos deben contener una animación capaz de permitir la diferencia clara entre un obstáculo corriente y un obstáculo mágico.

Para realizar cualquier tipo de animación en 3DS MAX se debe recurrir al sistema de partículas, sin embargo, Unity también posee un sistema de partículas muy similar. Además, estos sistemas de partículas suelen tener problemas con la exportación del 3DS MAX y se pierden algunas animaciones al ser importados en Unity. Por tanto, en 3DS MAX se probarán sistemas de partículas con plugins y en Unity sin plugins.

1) Animación con 3DS Max

La principal animación que se desea crear en el obstáculo será la de humo para generar la sensación de polvos mágicos. Existe gran cantidad de plugins que permiten la creación de humo, en concreto se ha decidido utilizar el plugin “Phoenix FD” de dinámicas de fluidos.

La mayoría de este tipo de plugins utilizan 3 componentes primordiales para la generación de humo. La denotación de cada componente varía en función del plugin utilizado:

- Foco de calor: Es el área donde se generará el humo en función de la intensidad de la fuente de calor.
- Fuente de calor: Es la batería que se utiliza para la creación de calor gracias a un valor de intensidad ajustable.
- Viento: Es el vendaval con una fuerza específica que se formará desde una posición determinada hacia una dirección y sentido concreto.

El funcionamiento es el siguiente:

1º) Colocación del foco de calor.

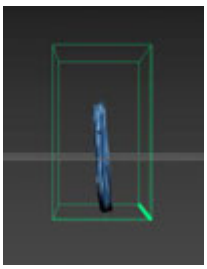


Fig.65 – Foco de calor en obstáculo con Phoenix FD.

Puesto que el objetivo es que el obstáculo obtenga efecto de humo, el área donde se creará el humo debe contener al obstáculo. Al no existir fuente de calor, no hay intensidad suficiente para la creación de humo.

2º) Creación de la fuente de calor.

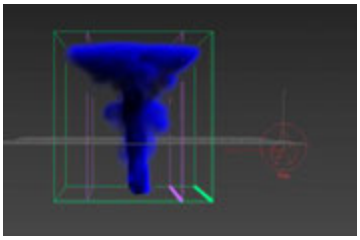


Fig.66 – Fuente de calor en obstáculo con Phoenix FD.

Con la intensidad adecuada de la fuente de calor se crea el humo en el foco de calor, pero se propaga hacia arriba.

3º) Posicionamiento del viento

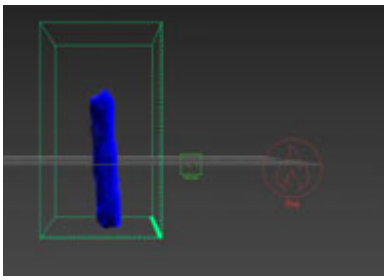


Fig.67 – Viento en obstáculo con Phoenix FD.

Si se crea un vendaval con la dirección paralela al suelo y sentido trasero del obstáculo, se consigue que el humo se propague paralelamente al suelo en vez de hacia arriba.

La cantidad de humo a generar dentro del obstáculo debe ser la adecuada para apreciarse el efecto. Hay que tener cuidado con generar excesiva cantidad de humo, ya que, dificultaría la visión de la parte trasera y podría darse el caso de que un obstáculo oculte a otro obstáculo que se encuentre detrás del mismo.

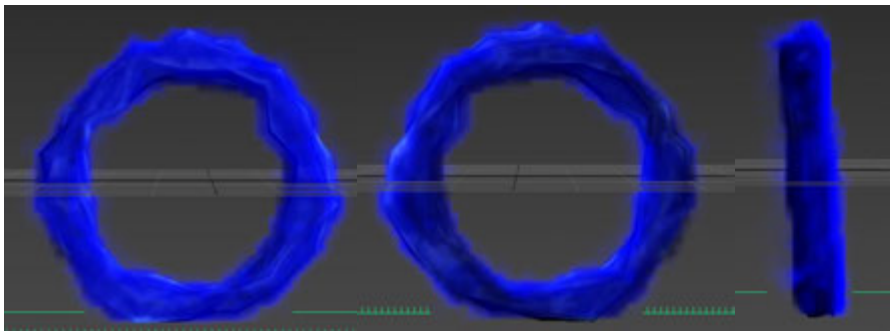


Fig.68 – Vistas del obstáculo esfera azul animada con Phoenix FD: frente, trasero y lateral izquierdo.

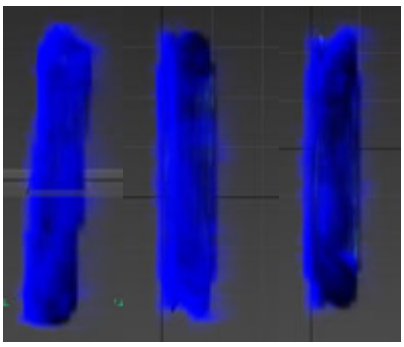


Fig.69 – Vistas del obstáculo esfera azul animada con Phoenix FD: lateral derecho, arriba, abajo.

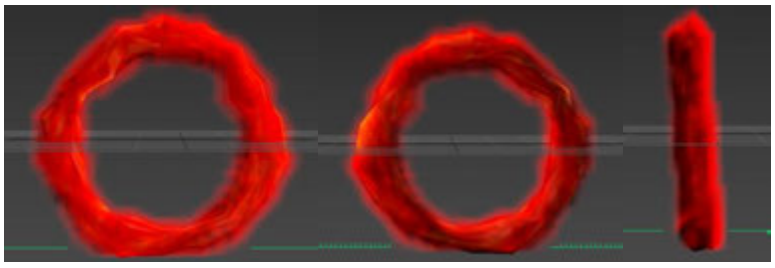


Fig.70 – Vistas del obstáculo esfera roja animada con Phoenix FD: frente, trasero y lateral izquierdo.

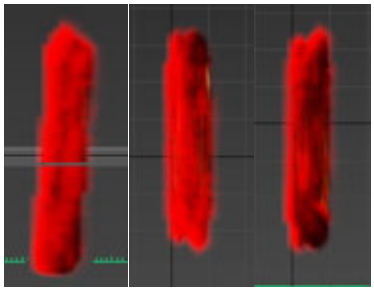


Fig.71 – Vistas del obstáculo esfera azul animada con Phoenix FD: lateral derecho, arriba, abajo.

2) Animación con Unity

Los sistemas de partículas permiten ajustar las propiedades de cada partícula dentro de un conjunto de partículas para generar una animación determinada. Por tanto, con la agrupación del conjunto de partículas se obtendrá una asociación de animaciones.

A continuación, se va a explicar el proceso de creación de cada animación.

1) Aro rotatorio

Esta animación pretende simular un aro que rotará continuamente a una velocidad determinada sobre sí mismo, aumentará y disminuirá ligeramente su tamaño para dar aspecto de movimiento y variará ligeramente de tonalidad de color.

Para ajustar el tamaño del aro, primero se define un rango de tamaño inicial. De esta forma, el aro que se crea no siempre tendrá las mismas dimensiones. También, se ajusta la variación del tamaño del aro que adoptará durante su tiempo de vida, con esto se consigue generar una sensación de movimiento.

Para crear la rotación del aro sobre sí mismo, se elige la velocidad del ángulo de rotación.

Para crear la variación de tonalidad de color, se puede modificar el gradiente del color que adopta durante distintas etapas de su tiempo de vida.

Este comportamiento se mantiene en forma de bucle. Finalmente se elige la cantidad máxima de partículas que pueden estar existiendo en cada instante, la duración de vida de cada una y la cantidad de partículas nuevas que se forman cada cierto tiempo.

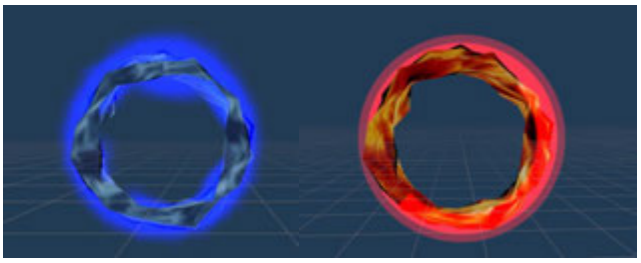


Fig.72 – Un arco rotatorio en obstáculo con Unity.

Se aprovechará este mismo sistema de partículas creado para insertar en el interior del obstáculo más sistemas de partículas con el mismo funcionamiento, pero escalado al interior.

Primero se muestra la inserción de dos arcos rotatorios.

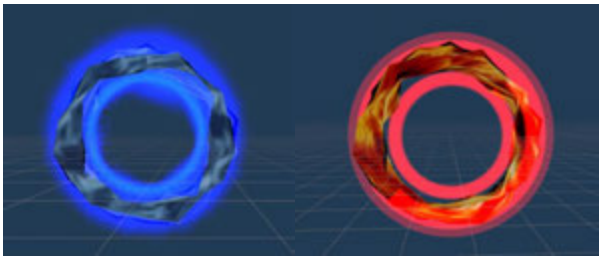


Fig.73 – Dos arcos rotatorios en obstáculo con Unity.

Después se muestra la inserción de tres arcos rotatorios.

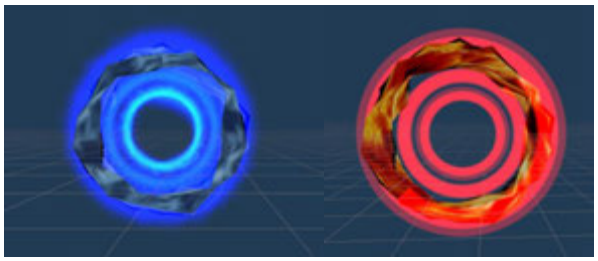


Fig.74 – Tres arcos rotatorios en obstáculo con Unity.

Gracias a estos sistemas de partículas se facilita al jugador encontrar la posición central del obstáculo. Además, por cada sistema de partículas insertado en el interior se ha aumentado la tonalidad del color para distinguir con mayor sencillez cada uno.

2) Núcleo

Esta animación pretende simular una bola central en el espacio vacío del obstáculo que aumentará y disminuirá continuamente su tamaño y variará ligeramente de tonalidad de color.

Para obtener este sistema de partículas se ha utilizado la misma técnica explicada anteriormente con el aro rotatorio.

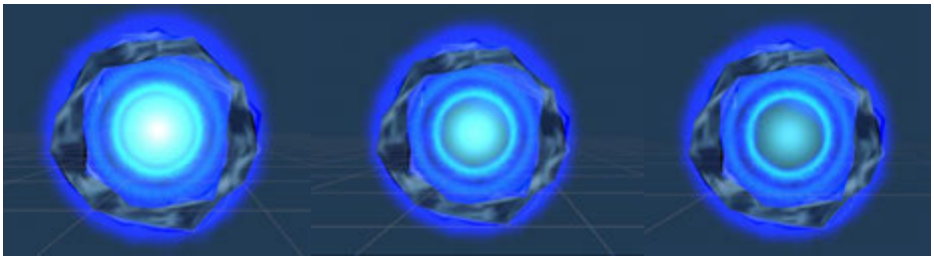


Fig.75 – Núcleo en obstáculo azul con Unity

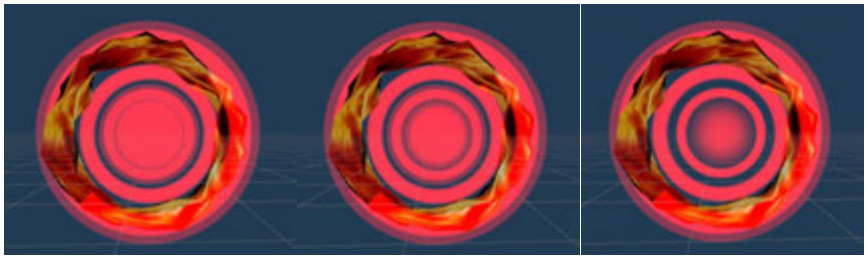


Fig.76 – Núcleo en obstáculo rojo con Unity

La tonalidad de la bola central es más clara que los aros para divisar el centro del obstáculo. Además, la bola varía entre un rango elevado de tamaños para que el movimiento de crecimiento y decrecimiento permita deducir el centro del obstáculo.

3) Humo

Esta animación pretende simular un humo con movimiento alrededor del obstáculo.

Para crear un desplazamiento del humo, se define un rango de velocidad. Se ha decidido utilizar un rango de velocidades bajo porque el humo sirve de acompañante a

la totalidad de los sistemas de partículas. Si por el contrario se utiliza velocidades altas, el humo adquirirá mayor atención que el resto de los sistemas.

Para este caso, la cantidad de partículas máximas que pueden existir en cada instante afectará al volumen del vapor total del humo. Como no se pretende ocultar el obstáculo entre el humo, la cantidad de partículas será escasa.

Para controlar el tamaño del área sobre el que se desplazará el humo, se utiliza un valor límite del campo de emisión. Este valor será ligero para evitar que se desplace el humo excesivamente atrás o delante y no disipe la visión del jugador.

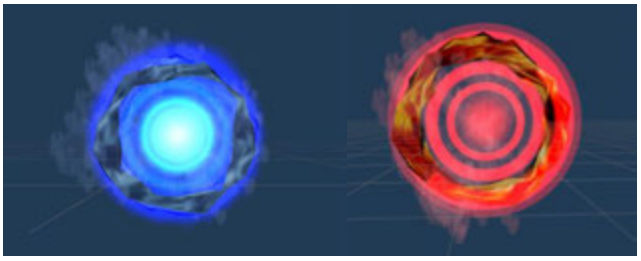


Fig.77 -- Humo en obstáculo con Unity

4) Espuma

Esta animación pretende simular la emisión de una serie de micropartículas con forma esférica alrededor del obstáculo.

El tamaño de las partículas variará en un rango mínimo de valores, ya que, su utilidad es generar la sensación de emanación de partículas desde la bola central.

Para otorgar movimiento y velocidad sobre las partículas se utilizaron las mismas técnicas ya explicadas en los sistemas de partículas anteriores.

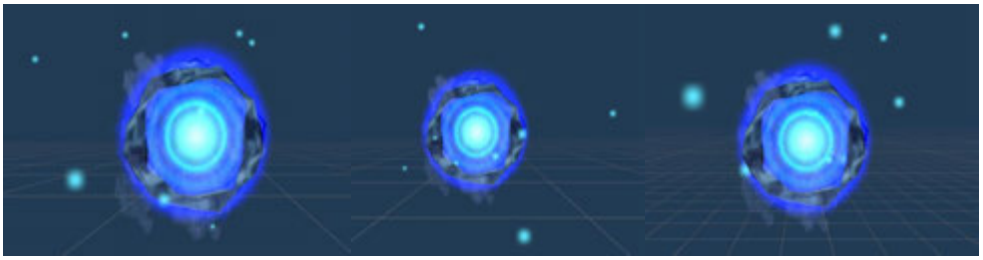


Fig.78 – Espuma en obstáculo azul con Unity.

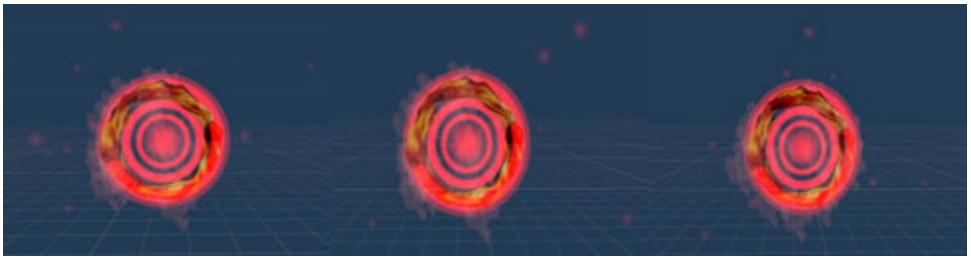


Fig.79 – Espuma en obstáculo rojo con Unity.

Se puede observar el resultado final con el conjunto de todos los sistemas de partículas. Este es el obstáculo que se utilizará.

Obstáculo funcional: Manos

De acuerdo con la temática de magia, lo más adecuado para las manos sería una varita mágica.

Clásicamente las varitas mágicas siguen un diseño de cilindro a modo de palo.

Fig.80 – Varita mágica clásica [102].

Posteriormente, empezaron a diseñarse otros modelos de varitas para diferenciar la parte donde se agarra con las manos. Por una parte, se empezaron a crear mangos para la comodidad de las manos. Por otra parte, se insertaron figuras decorativas en la punta para indicar el final de la varita.

Fig.81 – Mejoras de varitas mágicas clásicas [103] y [104].

A raíz de esto, empezaron a diseñarse combinaciones de las dos ideas y a crear prototipos muy variados de varitas.

Fig.82 – Varitas mágicas con mayor complejidad [105].

Teniendo en cuenta la diversidad de forma que puede adquirir una varita, se va a crear un diseño propio de varita mágica. La idea principal que se seguirá es la forma cilíndrica y la creación de un mango.

Modelado de las manos

El proceso de creación del modelado de la varita mágica va a ser dividido por partes:

1) Mango

Se trata de la parte inferior de la varita. Se utilizarán 8 cilindros agrupados en forma de redondeada para no perder la forma de base cilíndrica.

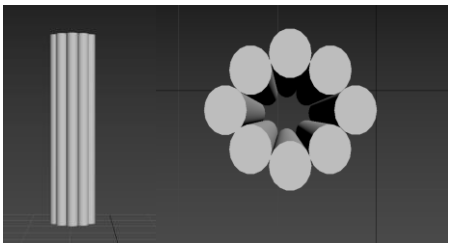


Fig.83 – Mago de varita mágica con cilindros rectos.

Existe una propiedad que permite retorcer un conjunto de objetos. Esta propiedad se utilizará sobre los 8 cilindros para perder el aspecto lineal que ofrece el conjunto pero sin escaparse de la forma de base cilíndrica propia de una varita.

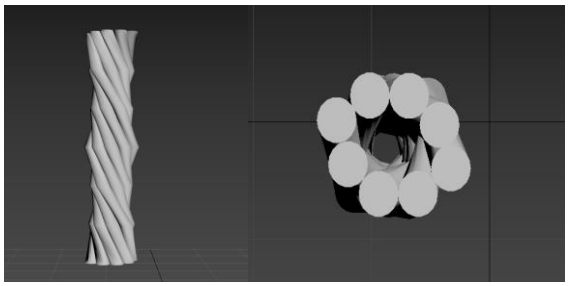


Fig.84 – Mago de varita mágica con cilindros retorcidos.

Se puede apreciar la forma resultante del mango, sin embargo, sería recomendable crear una señal que indique la superficie sobre la que debería apoyarse la mano. Para ello, se van a insertar toroides en el mango.

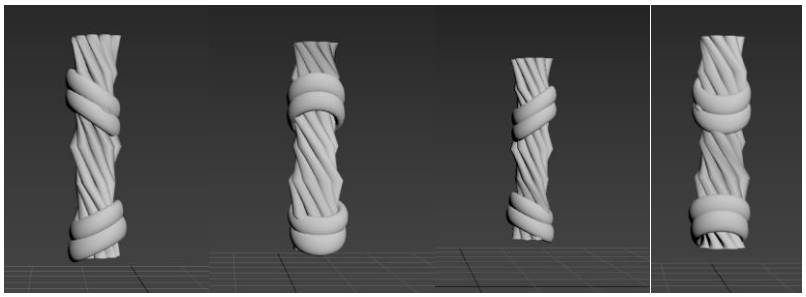


Fig.85 – Mago de varita mágica con cilindros retorcidos y toroides.

2) Decorativo

Se trata de la parte superior que conecta con el mango. Se utilizará un cilindro transformado en “Editable Poly”. Esta transformación hace que un objeto se convierta en un malla poligonal con varios niveles de sub-objetos.

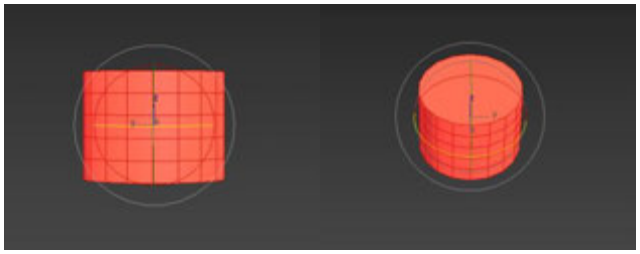


Fig.86 – Cilindro transformado en “Editable Poly”.

Se aumentará la cantidad de niveles de subjetos para hacer más largo el cilindro. Esto permitirá ser más preciso en la modificación de niveles concretos y así conseguir mejores acabados.

En los niveles deseados se va a aplicar un estrechamiento para que se genere un cilindro con zonas lineales con aumentos y disminuciones de grosor. Además, se eliminarán algunos niveles obteniendo trozos de cilindro flotantes.

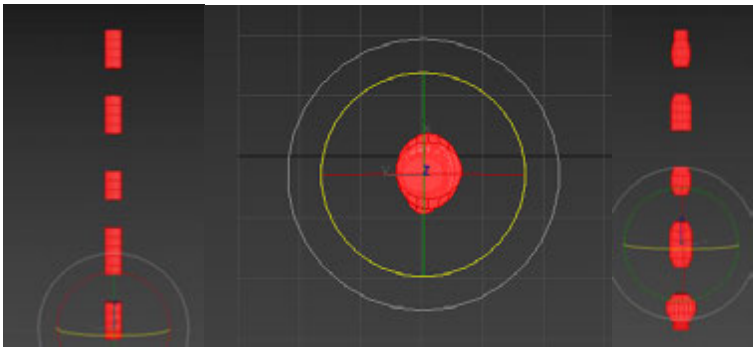


Fig.87 – Subdivisión de niveles de cilindro “Editable Poly”.

Los espacios vacíos entre los niveles del cilindro, serán rellenados por conjuntos de cilindros retorcidos entre sí (obtenidos con la misma técnica aplicada para el mango).

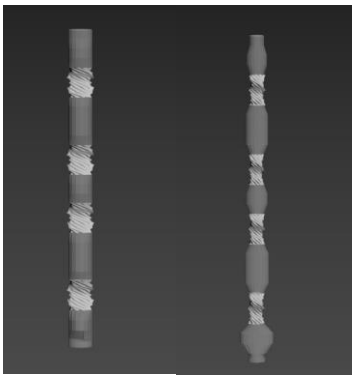


Fig.88 – Unión de subniveles de cilindro “Editable Poly” con conjunto de cilindros retorcidos.

Para que la varita termine con forma redondeada se insertará una esfera ajustada en la punta.

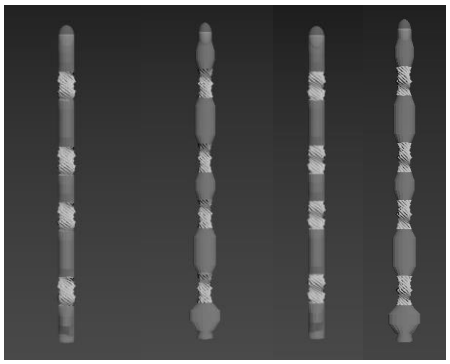


Fig.89 – Modelado de la parte superior de la varita mágica.

3) Fusión

Consiste en la incorporación del decorativo sobre el mango modelado anterormente.

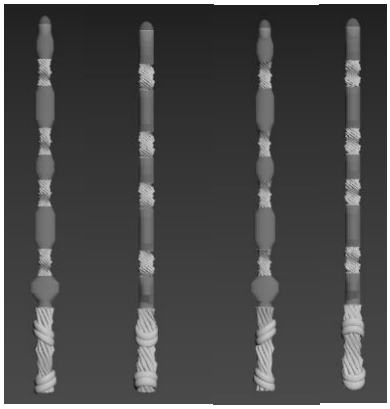


Fig.90 – Modelado resultante de la varita mágica.

Finalmente se ha obenido un modelado de varita mágica que mantiene la forma base de cilindro.

Texturizado de las manos

De la misma forma que los obstáculos, existirá una varita mágica azul y otra roja. Típicamente las varitas son de madera, por lo que las texturas seleccionadas mantendrán ese aspecto de madera.

Para la varita azul se ha decidido utilizar una textura de madera azul oscuro para los cilindros retorcidos y madera azul clara para la esfera de la punta. Para los niveles de la malla del cilindro superior se utilizará un azul más claro acristalado para desentonar con respecto al mango.

Fig.91 – Texturas utilizadas para la varita azul [100].

Fig.92 – Modelado y texturizado de varita azul.

De la misma forma, para la varita roja se ha decidido utilizar una textura de madera roja oscura para los cilindros retorcidos y madera roja clara para la esfera de la

punta. Para los niveles de la malla del cilindro superior se utilizará un rojo más claro acristalado.

Fig.93 – Texturas utilizadas para la varita roja [100].

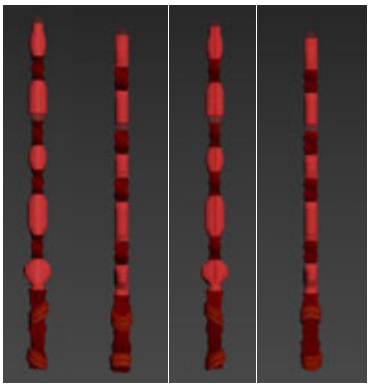


Fig.94 – Modelado y texturizado de varita roja.

Animación de las manos

La varita estará presente en todo momento al jugador durante la partida. Por tanto, no se debe realizar una excesiva cantidad de efectos porque las varitas deberán actuar de forma desapercibida. El propósito es crear un escenario amigable para el jugador y no una continua distracción.

El efecto que se utilizará será similar al efecto de la bola central del obstáculo esfera. Sin embargo, no se variará el tamaño de la bola para que actúe siempre con un tamaño constante. Así el usuario se adaptará rápidamente a la varita.

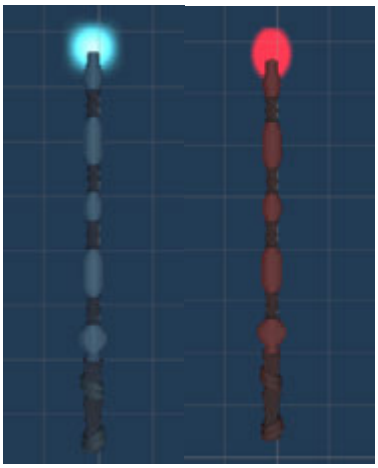


Fig.95 – Varita azul y roja animada.

Árbol

En un bosque comarcal puede haber distintos tipos de árboles en función de la naturaleza propia de la zona.

La belleza de un escenario no se mide por los objetos individuales en sí, si no por el ambiente generado por la totalidad que conforman. La elección de un árbol no debe basarse únicamente en su modelado final si no en el entorno donde se desarrolla.

Teniendo en cuenta el entorno generado anteriormente, serán más adecuado utilizar pinos o las encinas de menor tamaño.

Fig.96 – Árboles adecuados al entorno: pino [106] y encina [107].

Debido a que se pretende utilizar una gran cantidad de árboles para distinguir el camino por donde circularán los obstáculos, será recomendable utilizar pinos. Ya que, las encinas agrupadas pueden generar mayor cantidad de irregularidades sobre el terreno.

Modelado del árbol

El pino se dividirá en diversas partes:

1) Tronco

Se empezará con el modelado del tronco. El tronco vendrá dado principalmente por un cilindro.

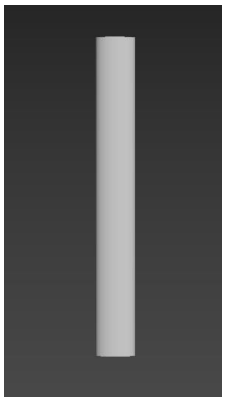


Fig.97 – Modelado inicial del tronco del pino.

Para indicar la copa del árbol, el cilindro se convertirá a “Editable Poly” para hacer que el nivel superior de la malla sea más estrecho.

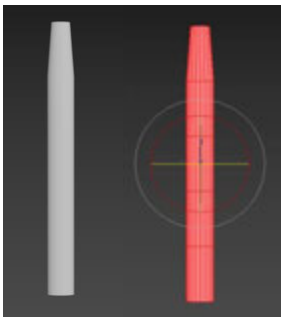


Fig.98 – Modelado resultante del tronco del pino.

2) Ramas

Las ramas se crearán mediante líneas de dos dimensiones. Para ello, se definirán las coordenadas bidimensionales de los puntos que se unirán a través de una recta.

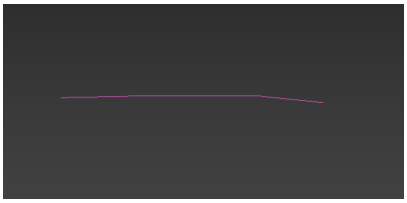


Fig.99 – Rama bidimensional del pino.

Para convertir la línea bidimensional en tridimensional, se pueden modificar los ajustes de renderización de la línea 2D y así asignarle profundidad.

Fig.100 – Rama tridimensional del pino.

Para crear una hoja del pino, se va a crear un prisma rectangular y se va a convertir en “Editable Poly”. Entonces se va a disminuir el tamaño de una base para que tenga una base una forma puntiaguda.

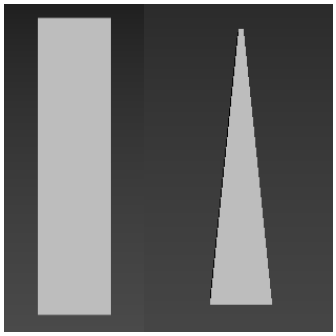


Fig.101 – Transformación de base superior de un rectángulo a un punto.

Para que tenga un formato más redondeado, los niveles del centro del rectangulo se van a ensanchar y así se definirá una curva.

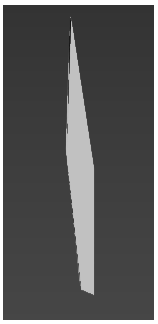


Fig.102 – Modelado de hoja de pino resultante: generación de curva en un rectángulo transformado a una base superior puntiaguda.

Este modelado de hoja se va a duplicar para posicionarse a lo largo de toda la rama creada anteriormente.

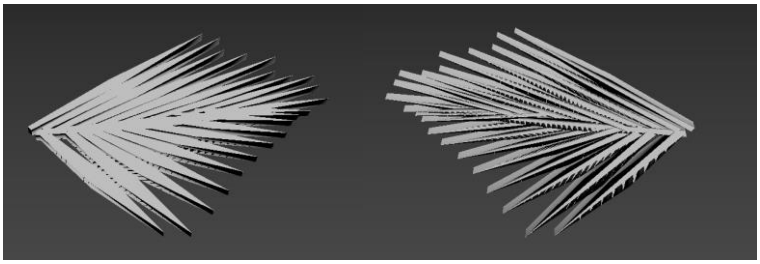


Fig.103 – Posicionamiento de la hoja sobre la rama tridimensional.

Tras obtener una rama poblada de hojas, se va a doblar todo el conjunto con la propiedad “bend” para dar aspecto de rama real.

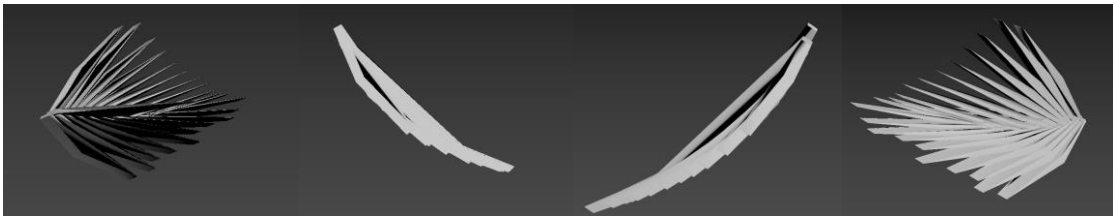


Fig.104 – Modelo resultante de rama poblada con hojas del pino.

Para generar el resto de las ramas, se realizarán duplicados con diferentes escalas de la primera rama modelada y se posicionarán a lo largo del tronco creado inicialmente.

Fig.105 – Modelo resultante de pino.

Texturizado del árbol

Para texturizar el pino se van a elegir 3 colores diferentes para el pino y para el tronco.

Fig.106 – Texturas de hojas del pino [100].

Fig.107 – Texturas del tronco del pino [100].

Combinando estas texturas, se han obtenido 3 pinos aparentemente diferentes:

Fig.108 – Primer modelo texturizado del pino.

Fig.109 – Segundo modelo texturizado del pino.

Fig.110 – Tercer modelo texturizado del pino.

Los 3 pinos se posicionarán sobre el entorno para crear mayor variedad en los árboles.

Fig.111 – Vista del entorno con árboles en la posición delantera del jugador.

Al posicionarse los pinos sobre el entorno, el centro se mantiene despejado para dar a entender el recorrido por el que circularán los obstáculos. Además, estos pinos en el centro pueden obstaculizar el paso de los obstáculos.

Plantar una cantidad elevada de árboles puede influir negativamente en la fluidez del videojuego. Esto hace que se haya tenido que recurrir a técnicas visuales para dar sensación de elevada cuantía de pinos. En concreto, en el fondo delantero del usuario, se han posicionado pocos árboles ensanchados en el eje x y rotados para dar la apariencia de un conjunto de árboles.

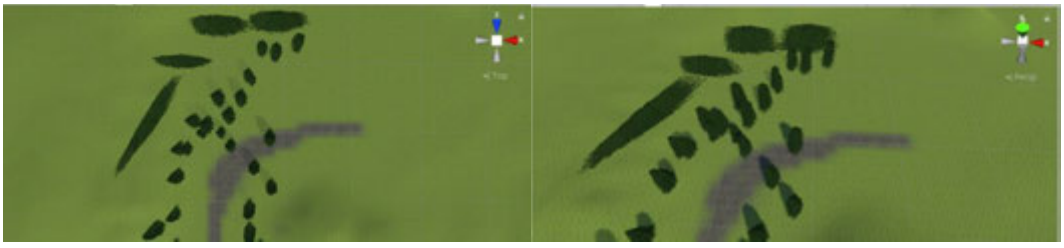


Fig.112 – Vista exterior del posicionamiento de los árboles frontales al jugador.

Gracias a esta técnica se consigue el mismo acabado que si hubiera una enorme cuantía de pinos.

Las perspectivas restantes del jugador no adquieren tanta importancia porque su concentración estará enfocada en la parte delantera. Es por ello, por lo que se han colocado árboles en menor medida, para que, si decide observar el resto del entorno, entienda que se encuentra en un bosque.

Fig.113 –Vistas de árboles del jugador: lateral derecho, lateral izquierdo y trasero.

Roca

Las rocas son un elemento principal que se encuentran presentes en cualquier tipo de entorno. Con las texturas que se aplicaron sobre el entorno se definió el camino rocoso. Por tanto, las rocas que se van a crear serán rocas grandes porque las pequeñas ya vienen integradas en el camino.

Con respecto a la forma, las rocas no siguen ningún patrón claro.

Fig.114 – Distintas formas de rocas reales [108], [109] y [110].

Las rocas pueden ser puntiagudas, redondeadas, estrechas, alargadas, engordadas… Las rocas que se evitarán crear serán las redondeadas puesto que son las que más se acercan a la simetría. A mayor cantidad de elementos simétricos dentro del escenario menor será el realismo alcanzado.

Modelado de la roca

Para modelar la roca se comenzará creando una geosfera.

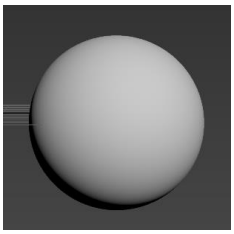


Fig.115 – Geosfera.

Aparentemente en 3DS MAX una geosfera y una esfera son similares. Sin embargo, internamente la geosfera está compuesta por mayor cantidad de segmentos que la esfera, lo que permitirá mayor precisión para moldearla.

Para deformar la roca, se utilizará la propiedad “displace” explicada anteriormente en el modelado del obstáculo esfera. Sin embargo, se incorporará también un mapeo de células que permitirá desplazar las deformaciones obtenidas por “displace” sobre los segmentos de la geosfera.

Se aplicará esta técnica para modelar tres rocas diferentes.

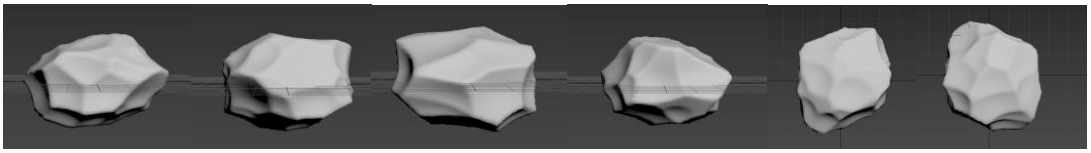


Fig.116 – Vistas del primer modelado de roca: frontal, lateral izquierdo, trasero, lateral derecho, encima, debajo.

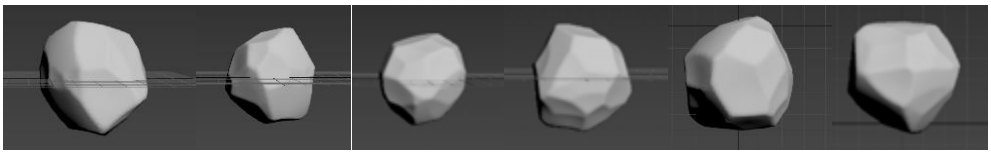


Fig.117 – Vistas del segundo modelado de roca: frontal, lateral izquierdo, trasero, lateral derecho, encima, debajo.

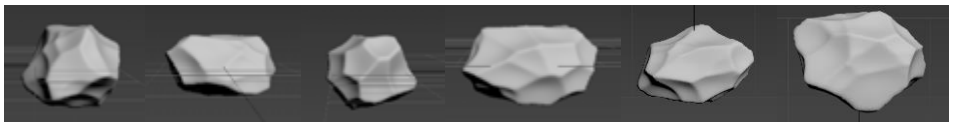


Fig.118 – Vistas del tercer modelado de roca: frontal, lateral izquierdo, trasero, lateral derecho, encima, debajo.

Se observa la no simetría en ninguna de las tres rocas.

Texturizado de la roca

Para texturizar las rocas se opta por unos colores simples pero realistas.

Fig.119 –Textura utilizada para las rocas [100].

Se aplicará la misma textura sobre las tres rocas.

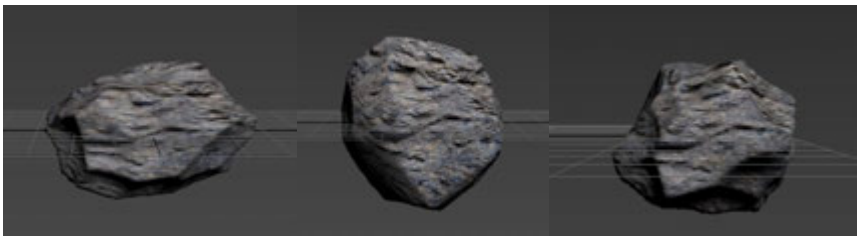


Fig.120 – Modelo y texturizado resultante de rocas.

Estas rocas se van a posicionar sobre el entorno, principalmente en la parte izquierda porque se aproxima a la montaña rocosa del fondo.

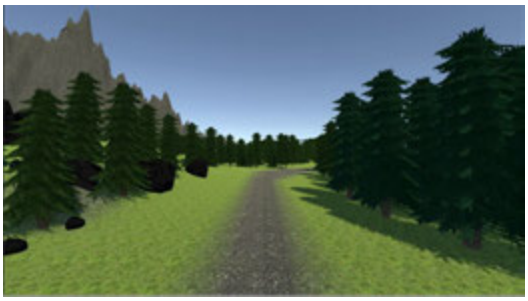


Fig.121 – Vista del entorno con árboles y rocas en la posición delantera del jugador.

Se consigue acompañar a los pinos con rocas de diversos tamaños. También se ha aplicado la técnica de ensanchamiento en el eje X sobre las rocas del fondo, para aparentar un cúmulo de rocas grandes.

Fig.122 – Vista exterior del posicionamiento de las rocas laterales al jugador.

Así, se consigue un aspecto rocoso reduciendo la cantidad de rocas.

Seta

Típicamente las setas mantienen el mismo patrón de forma. Contienen una base cilíndrica redondeada y una cabeza esférica.

Fig.123 – Distintos tipos de setas reales [111], [112] y [113].

Modelado de la seta

Para crear una seta se utilizará como figura base la esfera. Si se convierte la esfera en un conjunto de niveles de objetos gracias a “Editable Poly”, se puede desplazar los distintos niveles.

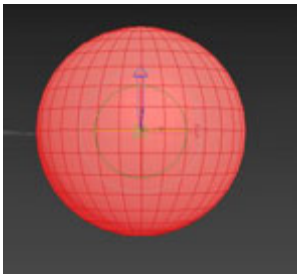


Fig.124 – Esfera convertida en "Editable Poly".

El nivel central se va a desplazar hacia arriba y se va a encoger para obtener el aspecto de la cabeza. Los niveles inferiores al centro se van a encoger y desplazar hacia abajo para conseguir una forma de base cilíndrica.

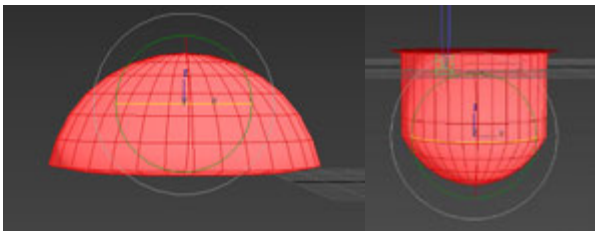


Fig.125 – Separación de subniveles de esfera “Editable Poly”.

Se obtendría la forma de seta resultante.

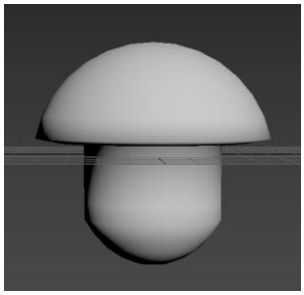


Fig.126 – Modelo resultante de seta.

Texturizado de la seta

La seta debe ser poco llamativa, por lo que, los colores utilizados deben ser oscuros o poco atractivos para el usuario.

Fig.127 – Texturas utilizadas para la seta [100].

Fig.128 – Modelo resultante de seta texturizada.

Las setas suelen crecer en zonas con humedad, poca luz o sombra de árboles. El entorno implementado está demasiado iluminado como para agregar altas cantidades

de setas. Por tanto, las setas se colocará alguna seta en huecos entre las rocas para embellecer el entorno.

Fig.129 – Vista del entorno con árboles, rocas y setas en la posición delantera del jugador.

Fig.130 – Vista exterior del posicionamiento de las setas laterales al jugador.

Flor

Las flores en un entorno suelen generar sensación de armonía y permite tranquilizar al usuario.

Las flores deben mantener concordancia con el entorno. Por tanto, no se deben modelar distintos tipos de flores si no poseen una relación entre sí.

El modelado de flor que se ha elegido es de la una margarita, ya que, esta flor tiene se presenta enormemente en gran variedad de terrenos.

Modelado de la flor

Para modelar el tallo de la flor, se utilizará una esfera y se accederá a la propiedad “Editable Poly” para desplazar los niveles inferiores del centro hacia abajo. También, se moverán algunos niveles inferiores sobre el eje X para generar curvas y evitar un tallo completamente recto.

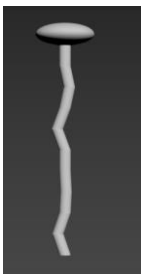


Fig.131 – Modelado resultante del tallo de la flor.

Para modelar las hojas se va a utilizar un plano y con las propiedades de “Editable Poly” se van a modificar las posiciones de los segmentos que conforman al plano. Así, se consigue asignarle una forma concreta.

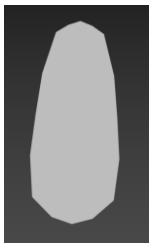


Fig.132 – Plano manipulado en “Editable Poly” con forma de pétalo.

Para doblar la hoja se utilizará la propiedad “FFD 3x3x3”. Esta propiedad permite controlar los puntos principales del plano. En este caso, hay 6 puntos principales: 2 superiores, 2 centrales y 2 inferiores. Si alguno de estos puntos es desplazado, la forma del plano se ajustará al desplazamiento, pero sin modificar la posición del resto de los puntos principales. Esto permite dar una posición concreta a cada punto principal y conseguir una curva sobre la hoja.

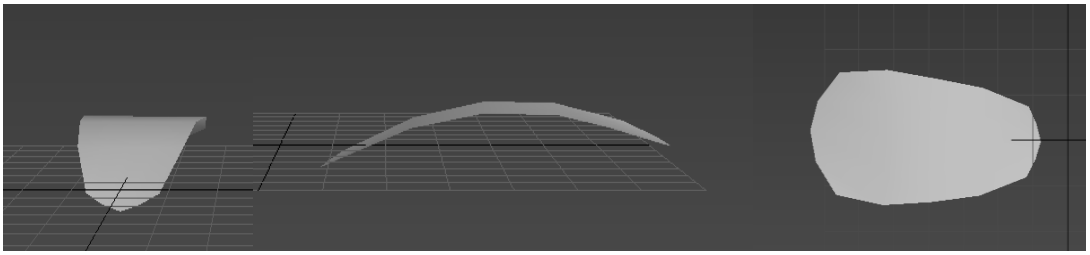


Fig.133 – Vistas del modelo de pétalo de flor resultante: frontal, lateral y por arriba.

La hoja resultante se duplicará y se posicionará sobre la cabeza del tallo.

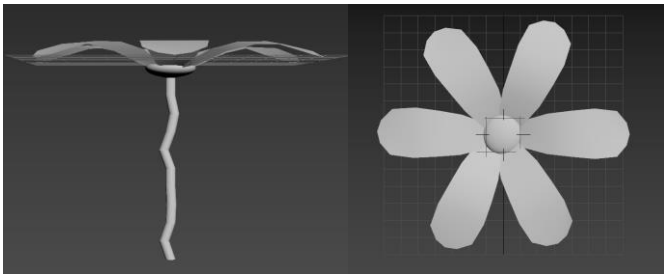


Fig.134 –Vistas del modelo resultante de flor: frontal y por arriba.

Se aprecia el acabado de los pétalos de la flor.

Texturizado de la flor

Para texturizar la flor se van a elegir diferentes colores para el pétalo, el tallo y la cabeza.

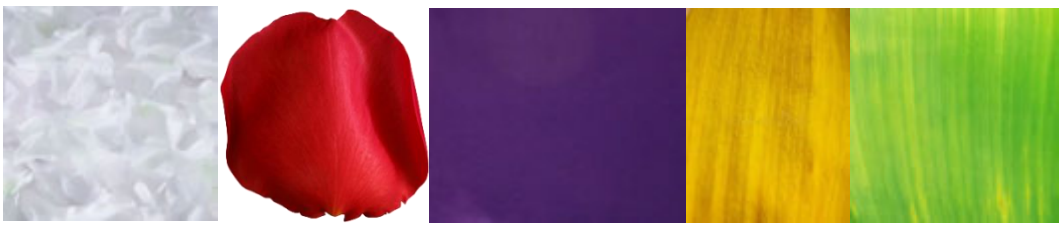


Fig.135 – Texturas utilizadas para las flores [100].

Estas texturas se combinarán para crear acabados diferentes de flor.

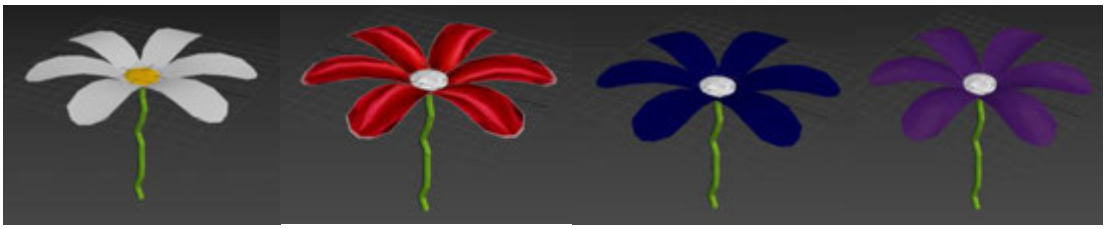


Fig.136 – Modelos texturizados resultantes de flor.

Las flores se posicionarán próximas a los árboles para aumentar la vegetación de la zona.

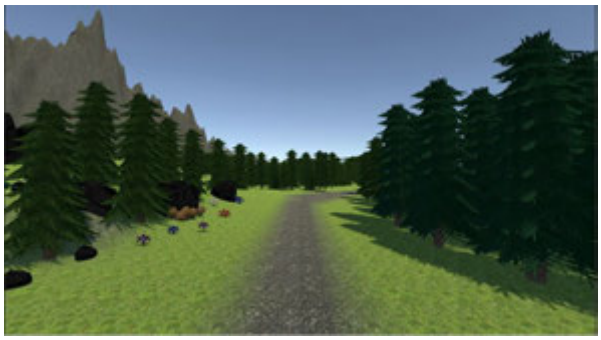


Fig.137 -- Vista del entorno con árboles, rocas, setas y flores en la posición delantera del jugador.

Conclusiones del escenario complejo

Finalmente, el resultado del escenario complejo es el siguiente:

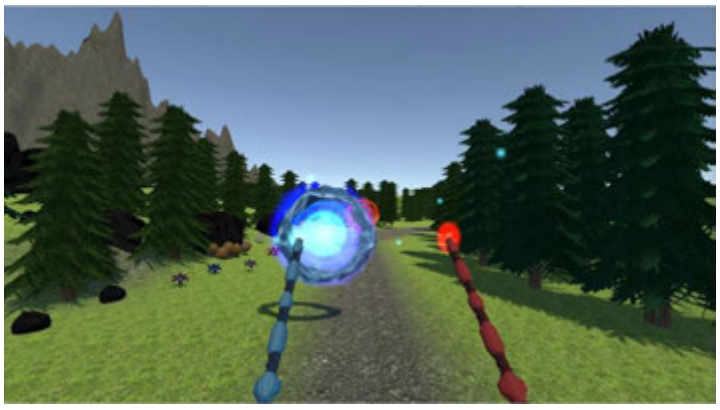


Fig.138 – Resultado del escenario complejo.

En un primer vistazo se puede apreciar el aspecto de bosque que representa. También, los efectos especiales utilizados para las esferas y las varitas hacen que termine con buen acabado el escenario y que demuestre al usuario el trabajo realizado en cada detalle.

Este escenario mantiene ligada una relación con el escenario simple, lo que permite al usuario saber qué es lo que tiene que hacer sin ningún tipo de problema tras haber jugado al escenario inicial.

3.3.4. Implementación del funcionamiento

3.3.4.1. Introducción

El propósito de esta sección es explicar la implementación de los objetos que deben ser funcionales, es decir, los obstáculos y las varitas mágicas. Primero se explicará las propiedades físicas que deben tener los objetos y, posteriormente la programación para controlar su comportamiento.

3.3.4.2. Físicas en objetos funcionales

En este apartado se explicará las propiedades físicas requeridas de los objetos funcionales del escenario complejo. Para el escenario simple se aplica la misma técnica, por tanto, no se explicará.

Obstáculos con forma de esfera

Se deberá incluir “RigidBody” para que se encuentre bajo las leyes de la física. Así podrá recibir fuerzas para dirigir su movimiento. Puesto que se pretende que el obstáculo se desplace en dirección paralela al suelo, se deberá desactivar las fuerzas gravitatorias.

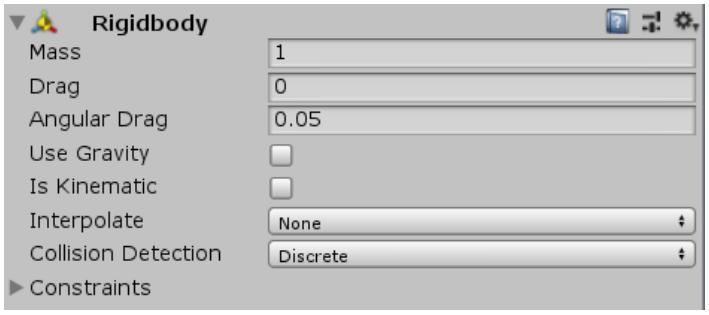


Fig.139 – Inserción de "RigidBody" a esfera sin fuerzas gravitacionales.

Se deberá de crear al menos un “collider” para permitir la colisión del obstáculo con el entorno. El “collider” se posicionará para definir el centro del obstáculo, por tanto, se tratará de una esfera con el mismo centro que el efecto de animación de la bola central. De esta manera, el centro de la esfera será reconocible.

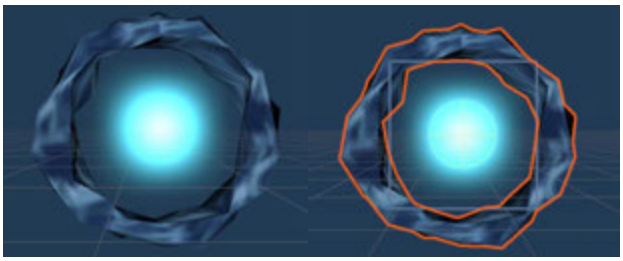


Fig.140 – Inserción de un “collider” con mismo radio y centro que bola central del obstáculo.

Tras haber ajustado el centro y el radio del “collider” esférico a la bola central del obstáculo, se puede aumentar el mismo “collider” para aumentar la zona sensible a colisiones del obstáculo.

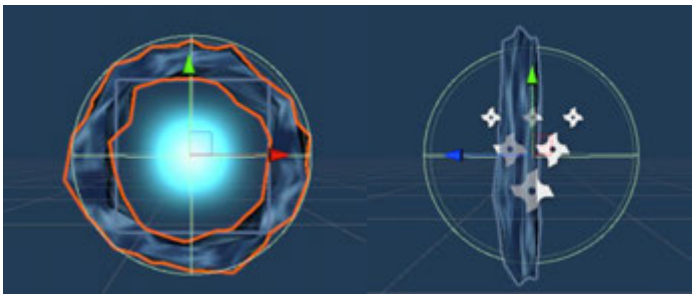


Fig.141 – Ampliación del “collider” esférico del obstáculo.

El problema con este tamaño es que el usuario tendría la posibilidad de tocar el obstáculo sin visualmente estar manteniendo un contacto con el mismo. Estos golpeos se alejarían de la intuición del jugador y entonces obtendría peores resultados. Para resolver este problema, se va a reducir la dimensión del “collider” esférico y se va a crear otro “collider” con forma de caja.

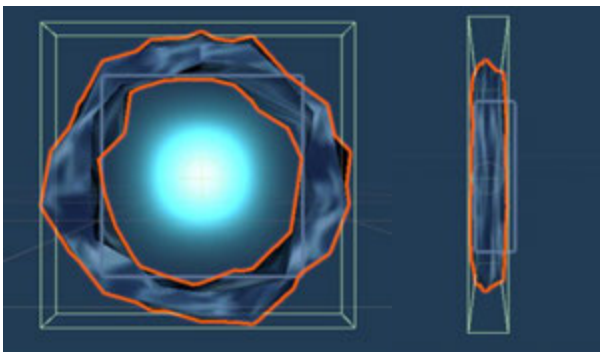


Fig.142 – Obstáculo azul con “collider” esférico y “collider” con forma de caja.

El “collider” con forma de caja permite que el usuario mantenga mayor zona de colisión y con la dimensión ajustada a la profundidad del obstáculo. Por tanto, el centro del obstáculo quedará definido por el “collider” esférico y el terreno de la colisión (externo al “collider” esférico) quedará definido por el “collider” con forma de caja.

Estas propiedades también se asignarán de la misma manera con el obstáculo rojo.

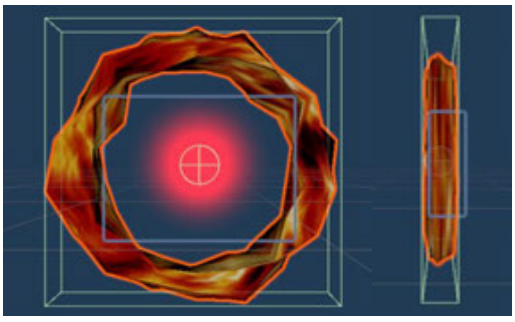


Fig.143 – Obstáculo rojo con “collider” esférico y “collider” con forma de caja.

Varitas mágicas

Las varitas mágicas tendrán que colisionar con los obstáculos, por tanto, necesitan de “Collider”. Se insertarán dos “collider”: uno en forma de caja para representar el modelo de varita mágica y otro en forma de esfera de acuerdo a la animación de bola que posee en la parte superior.

Fig.144 – Varitas mágicas con “collider” esférico y “collider” con forma de caja.

3.3.4.3. Programación del comportamiento

El comportamiento del sistema se programará en CSharp creando diferentes controladores.

Controlador de generación

Se trata del sistema de control principal de la partida. Las tareas primordiales que realiza son:

- Creación de obstáculo: las posiciones iniciales posibles del obstáculo vienen dadas por un parámetro que consiste en una lista de “Transformation”, es decir, una lista de posiciones tridimensionales. Los modelos del obstáculo de esfera posibles que se generarán estarán definidos en otro parámetro de una lista de “GameObject”. En concreto, existirán dos posiciones iniciales posibles de los obstáculos de esfera y dos modelos posibles del obstáculo (roja o azul).

  La selección de la posición inicial y el modelo del obstáculo será mediante un aleatorio entre cualquiera de los valores de las listas.
- Listar obstáculos: se utiliza una lista de “GameObject” que mantendrá el registro de cada obstáculo generado. Por tanto, se almacenará en el registro el obstáculo tras ser creado.
- Contador intervalo de tiempo entre obstáculos: existe una variable que sirve como cronómetro del tiempo sucedido desde la creación del último obstáculo. También se utiliza otra variable que indica el período de tiempo de espera hasta la creación del próximo obstáculo.

  El cronómetro vuelve a cero cada vez que se crea un nuevo obstáculo y se utiliza para permitir la creación de un obstáculo tras haber transcurrido el período de tiempo de espera de creación del próximo obstáculo.
- Finalizar partida: hay una variable que sirve como cronómetro del tiempo transcurrido desde que empezó la partida. También, existe una variable que indica la duración total de la partida.

  El cronómetro se utilizará para finalizar la partida tras haber transcurrido el tiempo total. Cuando se finaliza la partida, el registro de obstáculos mantiene únicamente los obstáculos no destruidos, es decir, aquellos que no ha alcanzado el jugador. Por tanto, se recorre esta lista de obstáculos para eliminarlos y cuantificar la cantidad de obstáculos no golpeados por el usuario.

El controlador de creación también contiene variables de tipo entero que se utilizan para almacenar: la cantidad de obstáculos azules y rojos destruidos, no destruidos y destruidos erróneamente y la distancia total al centro de los obstáculos

azules y rojos destruidos. Estos valores serán actualizados por el controlador de los obstáculos.

Al finalizar la partida, el controlador de creación crea un fichero de salida con un nombre definido por una variable y escribe los resultados obtenidos por el paciente. Si el fichero ya existe entonces escribirá los resultados al final del mismo fichero.

Controladores de obstáculos

Trata el sistema de control de los obstáculos esfera.

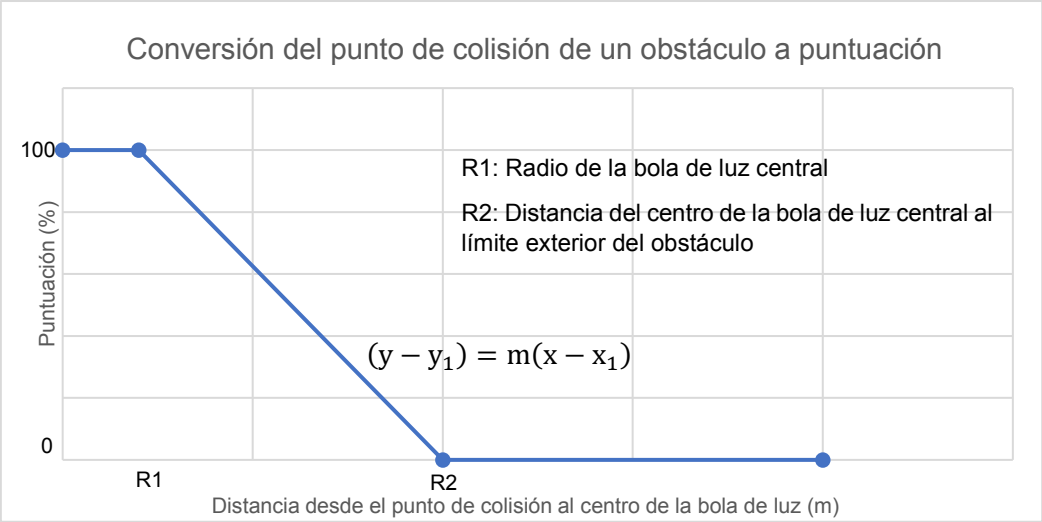
- Desplazamiento: Existe un parámetro que indica la velocidad de desplazamiento de los obstáculos. Para desplazar al obstáculo continuamente, se modifica su posición del eje Z en función de la velocidad dada por el parámetro. De esta forma, se consigue un desplazamiento únicamente de proximidad hacia el usuario.
- Destrucción: Unity contiene la función propia "OnCollisionEnter(Collision theCollision)" que se activa cuando un objeto detecta una colisión con cualquier elemento del entorno. En este caso, los obstáculos se desplazan por un camino libre y vacío, por lo que, la única colisión que detectarán será la colisión con las varitas mágicas.

  Por otra parte, Unity ofrece la posibilidad de nombrar con etiquetas a los objetos. Estas etiquetas no son únicas para cada objeto, es decir, varios objetos pueden tener la misma etiqueta. Sin embargo, para este caso se crearán dos etiquetas "varitaRoja" y "varitaAzul" y se asignará cada una a una a una varita mágica.

  Las etiquetas permiten que, al detectar una colisión, dependiendo de la etiqueta del objeto colindante se actúe de una determinada manera u otra.

  Para el caso del controlador del obstáculo azul, al detectar una colisión con un objeto con etiqueta "varitaAzul" se desarrollará el caso de que el usuario golpea un obstáculo con la varita correcta. Será entonces cuando se obtiene el punto de colisión de la varita con el obstáculo y el centro del obstáculo. Hay que tener en cuenta, que el usuario no puede acceder al centro del obstáculo, ya que, al existir profundidad en el obstáculo, la primera colisión se activará en el primer contacto de la varita con el obstáculo, es decir, el punto de colisión será en la capa más externa del obstáculo. Por tanto, sería imposible que el usuario golpeara en el punto exacto del centro. Para solucionar este problema, lo que se hace es convertir tanto el punto de colisión de la varita con el obstáculo como la posición del centro a un punto bidimensional. De esta forma, el usuario podría golpear el centro bidimensional del obstáculo. Para medir la precisión del golpeo, se utiliza la función integrada de Unity "Vector2.Distance(Vector2 origen, Vector2 destino)" que calcula la distancia entre dos puntos aplicando la distancia euclídea.

  Es importante remarcar que se considerará un golpe de precisión 100% si se encuentra a una distancia menor o igual al radio de la bola de luz central del obstáculo. Por otra parte, si el golpe se diera a partir del límite exterior del obstáculo (colisionaría con el "collider" en forma de caja) se considera una puntuación de 0%. Los golpes ubicados entre el exterior de la bola de luz y el límite exterior del obstáculo tendrán que realizar una conversión de puntuación. De esta forma, la puntuación seguirá la siguiente gráfica:



Gráfica 1. Conversión del punto de colisión de un obstáculo a puntuación.

Se puede observar la ecuación punto pendiente en la diagonal de la gráfica. Para despejar la puntuación de un golpe, se despeja la variable “y”.

$$y = m(x - x_1) + y_1 \qquad (3.1)$$

Sabiendo que la fórmula de la pendiente es:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{0 - 100}{R_2 - R_1} \qquad (3.2)$$

Sustituyendo, se obtiene la ecuación resultante:

$$y = \frac{-100}{R_2 - R_1}(x - R_1) + 100 \qquad (3.3)$$

Tras haber calculado la precisión, se incrementará al contador de precisiones de obstáculos azules del controlador de generación y se incrementará el contador de obstáculos azules destruidos.

Para destruir el obstáculo se accede a la posición de la lista de obstáculos del controlador de generación que referencie al obstáculo golpeado y se elimina de esa lista. Posteriormente, se destruye el objeto con “Destroy” para liberarse de la memoria y desaparecer de la vista del usuario.

Si por el contrario la colisión del obstáculo azul es con un objeto con etiqueta “varitaRoja”, se incrementa el contador de obstáculos azules destruidos incorrectamente del controlador de generación, se incrementa el contador de obstáculos azules destruidos y se contará como puntuación del 0%. Posteriormente, se elimina el obstáculo de forma similar que en el caso correcto.

Para el caso del controlador del obstáculo rojo, las colisiones funcionarán de la misma forma que el controlador del obstáculo azul, sin embargo, el golpeo con acierto sería en una colisión con un objeto etiquetado como “varitaRoja” y el golpeo erróneo sería en una colisión con un objeto etiquetado como “varitaAzul”.

3.3.4.4. Conclusiones

Con respecto a la funcionalidad, se ha conseguido establecer la conexión entre los obstáculos de esfera y el arma para destruirlos. Con respecto al diseño y al modelado de los objetos 3D, se han probado diferentes técnicas obteniéndose resultados de alta

de calidad. También se ha conseguido que estos objetos mantengan una relación entre sí y con el entorno desarrollado. En conjunto se ha conseguido crear un escenario complejo, pero con métodos de aprovechamiento de recursos lográndose así una escena embellecedora para el usuario.

3.3.5. Creación del menú principal

3.3.5.1. Introducción

El menú principal se mostrará como escenario inicial nada más cargar el videojuego. El propósito del menú principal es dar la posibilidad al usuario de modificar los parámetros básicos del sistema tales como velocidad de los obstáculos, intervalo de salida entre cada obstáculo y duración de la partida y la elección del escenario (escenario simple o complejo) donde utilizar dichos valores de parámetros.

A continuación, se mostrará el modelado del menú principal que se ha utilizado y la texturización y, posteriormente se explicará la interacción del usuario en el sistema.

3.3.5.2. Modelado y texturización

Es crucial que escenario principal sea lo más amigable y cómodo para el usuario, ya que, las primeras impresiones del videojuego vendrán dadas por este escenario.

Se ha decidido que el usuario pulse botones “físicamente” para modificar cada valor de cada parámetro. De esta forma, se explota desde el primer momento del videojuego la idea de realidad virtual y así, se consigue impresionar al usuario.

Para modificar los valores de los parámetros deberían de existir 2 botones (uno que incremente y otro que decremento) para cada parámetro. Por tanto, se tendrían 6 botones de modificación de parámetros y 2 botones para elegir un escenario, en total 8 botones. También debería aparecer una pantalla para indicar el valor de cada parámetro, por tanto, se tendrían 5 pantallas. Además, es importante que aparezca una pantalla de bienvenida con las instrucciones del uso del videojuego, por lo que, existirá otra pantalla principal más central que explique estas instrucciones.

Para que los botones se encuentren a una altura razonable para el usuario, se utilizarán soportes verticales. Para decorar la habitación donde se encontrará el usuario, se pueden utilizar luces, ventanas y una puerta detrás de la habitación.

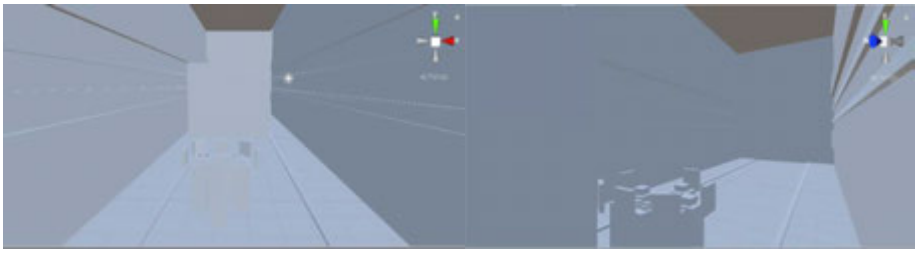


Fig.145 – Vista exterior del modelado del menú principal.

El usuario se encontrará en el centro entre las cinco pantallas. Se puede observar la forma alargada de la habitación. La dimensión de la habitación y el ambiente repleto de pantallas no genera un aspecto de habitación de una vivienda normal. Sin embargo, se puede utilizar este modelado resultante para crear el interior de una nave espacial.

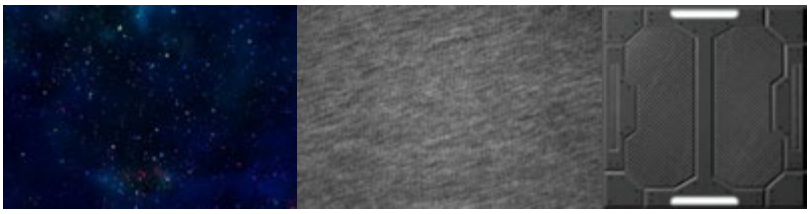


Fig.146 – Texturas utilizadas en el menú principal [100].

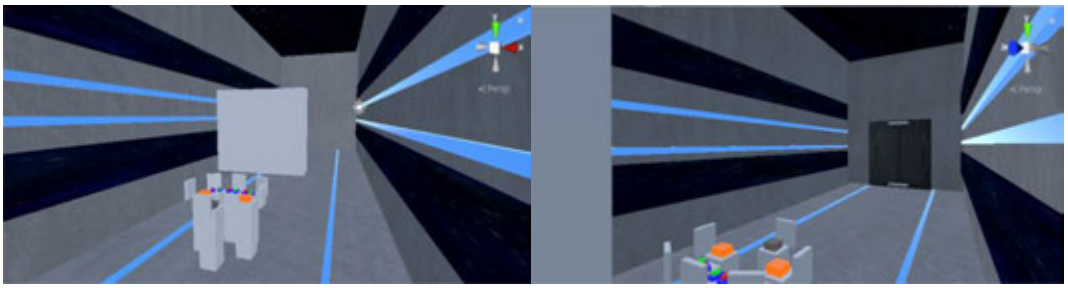


Fig.147 – Vista exterior del menú principal texturizado parcialmente.

Las paredes envueltas con el color grisáceo generan la sensación de paredes envueltas en papel decorativo, las ventanas y el techo con el fondo galáctico dan a entender que el usuario se encuentra en el espacio exterior, las luces azules sobre la habitación eliminan cualquier percepción de habitación normal y la puerta trasera es propia de las naves espaciales.

Respecto a los botones, se utilizará el color verde para los que incrementen el valor y el color rojo para los que decrementen el valor. El color naranja representará un botón que inicia la partida independientemente del escenario simple o complejo. El color grisáceo oscuro indicará el cierre del videojuego.

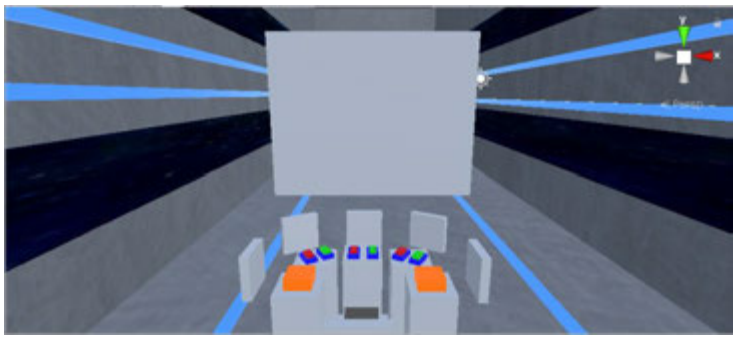


Fig.148 -- Vista exterior del menú principal texturizado.

Animación de la textura

Con el objetivo de crear sensación de movimiento a la nave espacial, se va a crear un desplazamiento sobre el eje X a las texturas galácticas. Para ello, existen dos parámetros dentro de la textura que deben ser modificados: el desfase para ajustar la correspondencia de la textura cada segundo y la posición principal de la ubicación de la textura para desplazarla en concordancia al desfase.

3.3.5.3. Funcionalidad de botones

El paquete de uso libre de “SteamVR” incluye scripts para ejercer funcionalidad a los botones. En concreto, se han utilizado dos scripts, “interactable.cs” se utiliza para hacer que un objeto sea interactuable por el usuario y “hover Button.cs” permite que un

objeto sea aplastable hasta un nivel concreto de altura, incluso permite que se generen funciones en algún período concreto de pulsación del botón.

Para hacer que una función modifique el contenido de un texto, se pasa el texto por parámetro y se accede a su contenido con la sentencia ".text". Entonces, se convierte la cadena de texto a entero con "int.Parse" y se comprueba si ese valor se puede incrementar o decrementar dentro de un rango preestablecido. Para el caso de la velocidad de los obstáculos y el intervalo de tiempo entre obstáculos el rango de valores es de 1-9 (incluidos) con incrementos y decrementos de una unidad y, para el caso de la duración de la prueba el rango es de [60,180] (incluidos) con incrementos y decrementos de 20 unidades.

Para hacer que una función cambie de escena se utiliza "SceneManager.LoadScene" junto con el nombre de la escena destino, sin embargo, antes de cambiar de escena es importante actualizar el valor de los parámetros seleccionados por el usuario en una instancia pública para que en la escena simple o compleja se puedan cargar esos valores previos del menú principal. También, es importante liberar de memoria al jugador antes de cambiar de escenas, ya que, si no es borrado existirán dos jugadores al volver a cada escena.

Para hacer que una función cierre un videojuego se utiliza la instrucción "Application.Quit".

Para hacer que un botón permita el cambio de escena, modificación de texto o cierre del videojuego, se ha insertado la función correspondiente al botón para que se active en el instante en el que el botón comienza a subir (después de haber sido pulsado hasta la altura máxima).

3.3.6. Interfaz de usuario

3.3.6.1. Introducción

En esta sección se van a explicar los elementos que se han insertado en Unity como elementos de interfaz de usuario. Primero se explicarán los elementos ubicados en el menú principal y, posteriormente los elementos ubicados en los escenarios de juego.

3.3.6.2. Menú principal

En el menú principal hay 7 pantallas que se muestran al usuario y deberán enseñar información en forma de texto. Para ello, en cada pantalla se crea un "canvas", es decir, una región poblada únicamente por elementos de tipo interfaz de usuario. Para que el texto dentro del "canvas" se extienda sobre toda la pantalla se utiliza la propiedad "World Space".

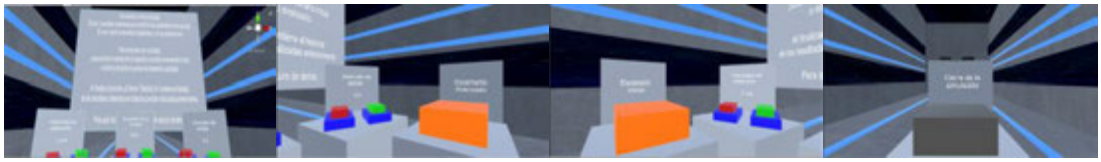


Fig.149 – Vista del usuario de las pantallas del menú principal: frontal, lateral derecho, lateral izquierdo y trasero.

Se puede contemplar cómo cada pantalla indica la información de forma clara al usuario. También, la pantalla principal ofrece toda la información esencial que necesitará el usuario para navegar por el videojuego.

3.3.6.3. Escenario simple y complejo

En el escenario simple y complejo se ha decidido mostrar la puntuación total de los obstáculos azules y rojos que lleva el usuario durante el transcurso de la partida. De esta forma, el usuario puede recibir información en tiempo real sobre si los golpes que está realizando mantienen elevada puntuación. Para ello, se insertan los textos utilizando "canvas" de la misma manera que en el menú principal. Para mantener actualizado el texto, se utilizará el contenido de la variable contador de precisiones de obstáculos azules y rojos del controlador de generación explicado anteriormente.

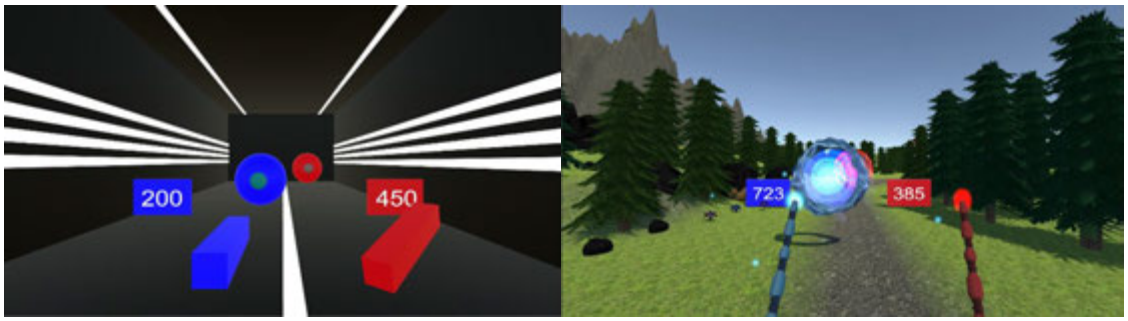


Fig.150 – Vistas del jugador con la información de la puntuación.

Por otra parte, para indicar el tiempo restante de la partida, se utilizará una barra horizontal que funcionará como un "sprite", es decir, una imagen de dos dimensiones. Además, se convertirá la barra en un "Sprite" rellenable para que, inicialmente la barra esté completamente llena y con el transcurso del tiempo la barra se vaya vaciando.

La cantidad de llenado de la barra se actualizará cada segundo con la función:

$$tiempo_{restante}/duración_{total} \qquad (3.4)$$

Las variables de la función se obtendrán utilizando las variables ya explicadas previamente del controlador de generación.

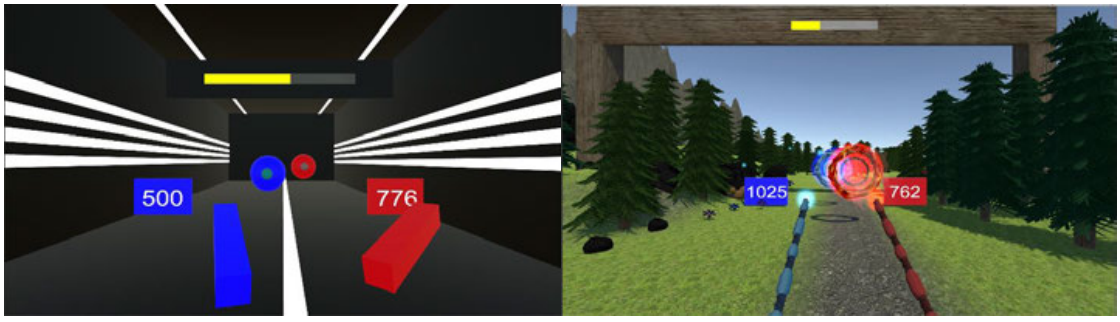


Fig.151 – Vistas del jugador con la información de la puntuación y del tiempo restante.

3.3.7. Inserción del sonido

La música en los videojuegos es un acompañante principal. El tipo de música utilizado viene muy influido por el tipo de videojuego desarrollado. Para este caso, se buscan canciones tranquilas porque el objetivo es relajar y mantener la concentración del paciente.

Basándose en los resultados más populares del buscador de YouTube, las canciones tranquilas utilizan principalmente pianos, guitarras y violines. Es por ello, por lo que se ha decidido utilizar una canción denominada “NO COPYRIGHT MUSIC - Gentle Trumpet Jazz” con el violín como instrumento protagonista para el menú principal. Por otra parte, se ha utilizado una canción denominada “Morning Routine by Ghostrifter - Chill - No Copyright Music” con el piano como instrumento protagonista para la escena simple y compleja.

Las canciones seleccionadas no tienen derechos de autor, son canciones que han subido los autores para su libre uso.

Para insertar una canción en Unity, se utiliza la propiedad “Audio Source” sobre los elementos que van a difundir el sonido y la propiedad de “Audio Listener” en el jugador para que pueda escuchar los sonidos emitidos.

Para insertar un sonido de destrucción a los obstáculos, se utilizará uno de los sonidos de defecto de Unity de explosiones. Para hacer que el sonido coincida con el momento de destrucción de un obstáculo, se colocará un “Audio Source” sobre las manos del usuario y, cuando alguna mano colisione con alguna esfera entonces el sonido comenzará a emitirse con la función “Play()”.

3.3.8. Optimización

La principal desventaja de la realidad virtual es su elevada consumición de recursos del sistema. Además, si se tiene un computador que se encuentra en el límite de los requisitos mínimos de realidad virtual (como es el caso del estudiante), la carga de los numerosos componentes supondrá un incremento desmesurado de la latencia y una disminución de la fluidez del videojuego.

En concreto, en el escenario del bosque, la cantidad de árboles influyen negativamente en la fluidez del videojuego, lo que supone un descontento en la experiencia del usuario. Es por ello por lo que se ha decidido llevar a cabo una optimización del escenario del bosque. Para ello, se va a reducir enormemente la cantidad de elementos decorativos, especialmente los árboles.

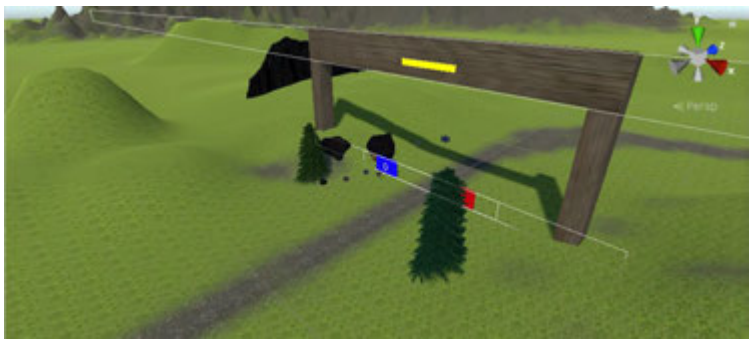


Fig.152 – Vista exterior del escenario bosque optimizado.

Con esta optimización también se consigue un escenario visualmente agradable al usuario y permite que mayor cantidad de jugadores puedan jugar y mejorar su experiencia.

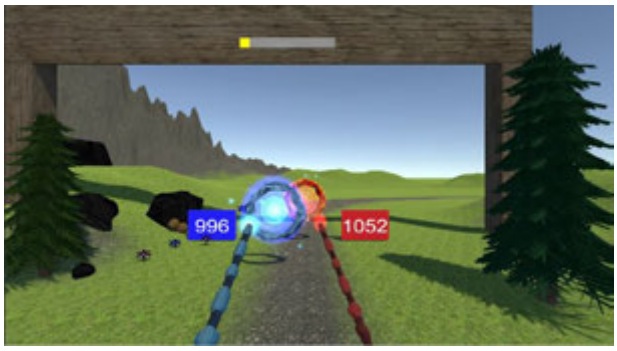


Fig.153 – Vista del jugador del escenario bosque optimizado.

Finalmente, se publicarán las dos versiones del videojuego: la optimizada y la no optimizada. Así, tanto los usuarios con recursos suficientes como los que dispongan de recursos computacionales más limitados, deberán de poder utilizar la aplicación con una fluidez suficiente.

4. EVALUACIÓN DEL SISTEMA

4.1. Introducción

El propósito de esta sección es evaluar el videojuego desarrollado en base a sus requisitos y de acuerdo a unas características determinadas. En esta sección se presentará el método de evaluación que se ha utilizad, los resultados y un balance y análisis de la evaluación alcanzada.

4.2. Explicación del método de evaluación

La metodología de evaluación del desarrollo del videojuego se basará en un sistema de puntos. Dicho sistema evalúa con una calificación del 1 al 10 cada uno de los objetivos compuestos por diversas tareas. Más concretamente, se puntúa cada tarea que compone a cada objetivo y la nota del objetivo será la media generada por las tareas que lo componen. Es posible que un objetivo obtenga una nota media superior a 10 debido al cumplimiento de tareas opcionales.

La transcripción de los números de cada tarea se realizará de la siguiente manera:

- De 0 a 5 puntos: tarea no superada.
- De 5 a 7 puntos: tarea superada con defectos.
- De 8 a 10 puntos: tarea superada gratamente.
- Más de 10 puntos: tarea superada satisfactoriamente y mejorada.

4.2.1. Evaluación de tareas y objetivos obligatorios

A continuación, se va a mostrar la puntuación asignada a cada tarea de cada objetivo:

Diseño y modelado de objetos tridimensionales:

- Obstáculo esfera: 10 puntos.
- Varita mágica: 10 puntos.
- Árbol: 10 puntos.
- Flor: 10 puntos.
- Roca: 10 puntos.
- Seta: 10 puntos.

Diseño y modelado del entorno:

- Montañas: 9 puntos.
- Colinas: 10 puntos.

Animación de los objetos tridimensionales:

- Obstáculo esfera: 10 puntos.
- Varita mágica: 10 puntos.

Exportación de modelos de objetos tridimensional de 3DS MAX:

- Correcta exportación de todas las figuras que componen a cada objeto: 10 puntos.
- Correcta inclusión de las texturas en cada objeto: 10 puntos.

Importación de modelos de objetos tridimensional a Unity:

- Importación de cada objeto manteniendo todas las características con la que se exportó: 10 puntos.

Efectos especiales visuales:

- Iluminación del entorno: 10 puntos.
- Efectos especiales de los obstáculos: 10 puntos.
- Efectos especiales de las varitas mágicas: 10 puntos.

Visión del jugador:

- Posibilidad de desplazamiento por el entorno: 10 puntos.
- Posibilidad de movilidad de las manos: 10 puntos.
- Posibilidad de girar la vista: 10 puntos.

Movimiento de los obstáculos:

- Movimiento en dirección paralela y sentido que se acerque al jugador: 10 puntos.

Interacción del usuario con los obstáculos:

- Movimiento en dirección paralela y sentido que se acerque al jugador: 10 puntos.

Fluidez del videojuego:

- Escenario simple: 10 puntos.
- Escenario complejo optimizado: 8 puntos.
- Escenario complejo no optimizado: 3 puntos.

4.2.2. Evaluación de tareas u objetivos opcionales

De forma separada, se muestran las tareas y objetivos opcionales realizados. Cada tarea opcional cumplida sumará un punto al objetivo.

Objetivo opcional: Efectos especiales sonoros.

- Incorporación de una música de ambiente: Suma 1 punto al objetivo.
- Incorporación de sonido al destruir cada obstáculo: Suma 1 punto al objetivo.

Objetivo obligatorio: Animación de los objetos tridimensionales. Se agregará una tarea opcional:

- Pruebas con plugins adicionales: Suma 1 punto a la media resultante.

4.3. Análisis de resultados

A continuación, se lleva a cabo la evaluación sobre la puntuación obtenida para cada objetivo del desarrollo del videojuego. Se representará en color verde aquellas tareas con una puntuación mayor o igual a 8 y en color rojo las tareas con puntuación inferior a 8.

4.3.1. Análisis de resultados de tareas y objetivos obligatorios

Diseño y modelado de objetos tridimensionales:

- Obstáculo esfera.
- Varita mágica.
- Árbol.
- Flor.
- Roca.
- Seta.

Media resultante del objetivo: 10 puntos.

Todos los modelos finales se han realizado desde 3DS MAX sin el uso de plugins y con acabados de calidad.

Diseño y modelado del entorno:

- Montañas
- Colinas

Media resultante del objetivo: 9.5 puntos.

El entorno se ha modelado con la capacidad suficiente para entender su representación tras pocos instantes de su visualización.

Animación de los objetos tridimensionales:

- Obstáculo esfera.
- Varita mágica.

Media resultante del objetivo: 11 puntos.

Se han probado diferentes tipos de animación en los objetos.

Exportación de modelos de objetos tridimensionales de 3DS MAX:

- Correcta exportación de todas las figuras que componen a cada objeto.
- Correcta inclusión de las texturas en cada objeto.

Media resultante del objetivo: 10 puntos.

Los objetos modelados en 3DS MAX se han exportado manteniendo la misma apariencia y estructura que en su respectivo proyecto de creación.

Importación de modelos de objetos tridimensionales a Unity:

- Importación de cada objeto manteniendo todas las características con la que se exportó.

Media resultante del objetivo: 10 puntos.

Los objetos exportados en 3DS MAX se han importado en Unity manteniendo la misma apariencia y estructura que en su respectivo proyecto de creación.

Efectos especiales visuales:

- Iluminación del entorno.
- Efectos especiales de los obstáculos.
- Efectos especiales de las varitas mágicas.

Media resultante del objetivo: 10 puntos.

Los objetos se han acompañado con diferentes tipos de efectos especiales sin entorpecer la visión del jugador ni disminuir la iluminación del entorno.

Visión del jugador:

- Posibilidad de desplazamiento por el entorno.
- Posibilidad de movilidad de las manos.
- Posibilidad de girar la vista.

Media resultante del objetivo: 10 puntos.

El jugador se puede desplazar por el entorno, mover las manos y girar la cabeza libremente para contemplar el entorno.

Movimiento de los obstáculos:

- Movimiento en dirección paralela y sentido que se acerque al jugador.

Media resultante del objetivo: 10 puntos.

Los obstáculos se acercan periódicamente al jugador.

Interacción del usuario con los obstáculos:

- Destrucción del obstáculo al colisionar con la varita.

Media resultante del objetivo: 10 puntos.

Los obstáculos son destruidos al colisionar con la mano.

Fluidez del videojuego:

- Escenario simple.
- Escenario complejo optimizado.
- Escenario complejo no optimizado.

Media resultante del objetivo: 7,33 puntos.

La fluidez del videojuego depende del equipo en el que esté siendo probado. El ordenador del estudiante permite jugar con total fluidez el escenario simple, con fluidez aceptable el escenario complejo optimizado y sin fluidez el escenario complejo no optimizado.

4.3.2. Análisis de resultados de tareas u objetivos opcionales

Objetivo opcional: Efectos especiales sonoros.

- Incorporación de una música de ambiente: Suma 1 punto al objetivo.
- Incorporación de sonido al destruir cada obstáculo: Suma 1 punto al objetivo.

Media resultante del objetivo: 2 puntos.

Se ha incorporado una música relajante de fondo.

Objetivo obligatorio: Animación de los objetos tridimensionales. Se agregará una tarea opcional:

- (Opcional) Pruebas con plugins adicionales: Suma 1 punto al objetivo.

Media resultante del objetivo: 11 puntos.

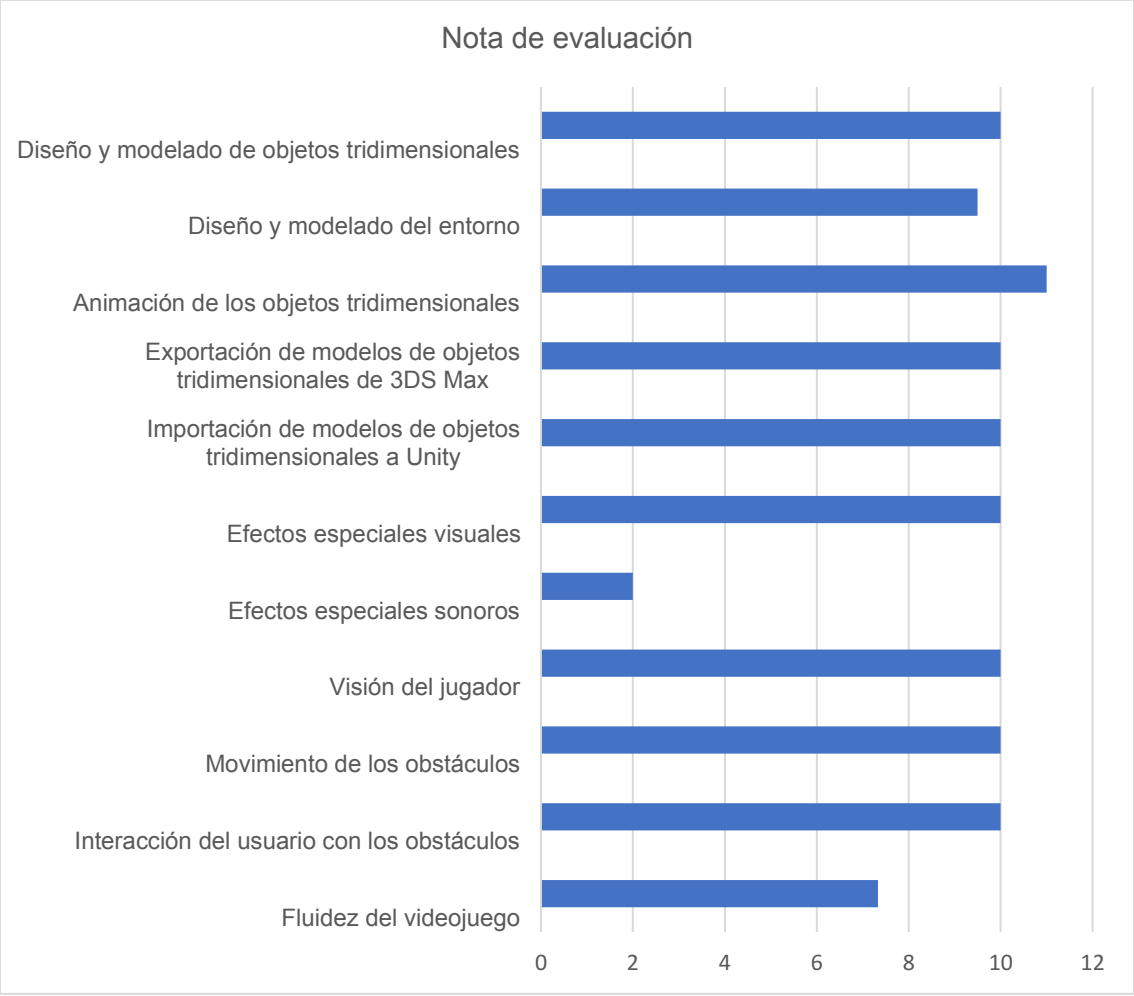
4.4. Puntuación media adquirida para la evaluación de los objetivos

Se va a mostrar la puntuación adquirida por cada objetivo en forma de tabla. Se agruparán tanto los objetivos opcionales como obligatorios.

TABLA 4.1. PUNTUACIÓN ADQUIRIDA PARA LA EVALUACIÓN DE LOS OBJETIVOS

Objetivo	Nota
Diseño y modelado de objetos tridimensionales	10
Diseño y modelado del entorno	9,5
Animación de los objetos tridimensionales	11
Exportación de modelos de objetos tridimensionales de 3DS MAX	10
Importación de modelos de objetos tridimensionales a Unity	10
Efectos especiales visuales	10
Efectos especiales sonoros	2
Visión del jugador	10
Movimiento de los obstáculos	10
Interacción del usuario con los obstáculos	10
Fluidez del videojuego	7,33

También se muestra la puntuación de los objetivos gráficamente:



Gráfica 2. Nota adquirida para la evaluación de los objetivos.

El objetivo sobre los efectos especiales sonoros ha obtenido 2 puntos porque es un objetivo opcional que se ha complido. La animación de los objetos tridimensionales se califica con 11 puntos debido al trabajo opcional de aprender a utilizar plugins para la animación de los objetos.

En general, todos los objetivos mantienen una nota superior a 9, esto significa que se han logrado satisfactoriamente.

La valoración total del videojuego desarrollado se mide como la media de las medias de las notas alcanzadas por los objetivos. El objetivo opcional indica un aumento de 1 unidad a la suma total, por tanto, la media se basará en 9 objetivos.

$$\text{Valoración total} = \frac{10 + 9.5 + 11 + 10 + 10 + 10 + 2 + 10 + 10 + 10 + 7{,}33}{10} = \frac{99{,}83}{10} = 9{,}983$$

4.4. Conclusiones

El sistema de evaluación utilizado permite obtener una nota numérica al proyecto con alto grado de precisión, ya que, permite dividir al sistema resultante en criterios independientes para encontrar fallos o mejoras en áreas concretas del videojuego.

Según el método de evaluación utilizado, el videojuego resultante cumple con los objetivos deseados y en algunos casos ha mejorado el resultado deseado.

5. PRUEBAS CON JUGADORES

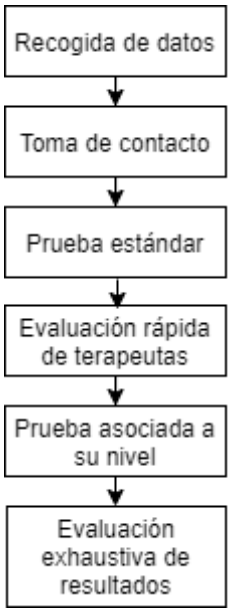
5.1. Introducción

El propósito de esta sección es mostrar el protocolo deseado para probar el videojuego con una serie de personas seleccionadas. Se mostrará la ficha descriptiva de cada jugador seguido de las condiciones de las pruebas a las que se han sometido y las conclusiones obtenidas en función de sus resultados alcanzados.

5.2. Protocolo

En primer lugar, la versión del videojuego seleccionada para realizar todas las pruebas con los jugadores es la optimizada. De esta forma, se asegura mayor fluidez del videojuego para todas las pruebas.

El procedimiento que se sigue en cada prueba es el siguiente:



1. Se registran los datos de utilidad del usuario.
2. El usuario juega una partida con condiciones estándar en el escenario simple y otra en el escenario complejo.
El propósito es que el jugador obtenga práctica y nociones sobre la base del videojuego, por lo que los resultados de estas partidas no se tendrán en cuenta.
3. EL usuario vuelve a jugar una partida con condiciones estándar en el escenario simple y complejo.
4. Los terapeutas analizan rápidamente los resultados obtenidos por el paciente y eligen nuevos valores de parámetros.
5. El usuario juega una nueva prueba en el escenario simple y complejo con los nuevos parámetros seleccionados por los terapeutas.
6. Los terapeutas analizan los resultados obtenidos por los pacientes.

Ilustración 154 - Esquema del protocolo de pruebas con personas

5.3. Condición de partida estándar

Todos los jugadores inicialmente realizarán partidas con condiciones estándar: duración de la prueba 100s, velocidad de obstáculos 3u/s e intervalo de tiempo entre cada obstáculo 3s.

Sin embargo, también se almacenan más condiciones de la prueba como la cantidad de obstáculos azules, rojos y totales, debido a la pseudoaleatoriedad implementada en el videojuego.

Recordatorio: los obstáculos azules se golpean con la mano izquierda y los rojos con la derecha.

5.4. Ficha descriptiva del jugador

Para la definición de un jugador se va a utilizar la siguiente ficha:

TABLA 5.1. FICHA DESCRIPTIVA GENÉRICA DEL JUGADOR.

Identificador	
Sexo	
Edad	
Mano dominante	
Discapacidad	
Experiencia en realidad virtual	
Equipo utilizado	

- Identificador: etiqueta única para distinguir cada jugador.
- Sexo: condición orgánica del jugador.
  - Hombre
  - Mujer
- Edad: cantidad de años del jugador el día en que se realizó la prueba.
- Mano dominante: mano con la que tiene mayor precisión el jugador.
  - Izquierda
  - Derecha
- Discapacidad: limitación de capacidad física o mental generada por una enfermedad neurológica.
- Experiencia en realidad virtual: si ya ha realizado ejercicios con realidad virtual previamente.
  - Sí
  - No
- Equipo utilizado: visores de realidad virtual utilizados junto con los complementos necesarios.

5.4. Análisis de resultados

5.4.1. Pruebas ajenas al protocolo

5.4.1.1 Jugador 1

Ficha del jugador 1

TABLA 5.2. FICHA DESCRIPTIVA DEL JUGADOR 1

Identificador	**P1**
Sexo	Hombre
Edad	62
Mano dominante	Izquierda
Diagnóstico principal	Traumatismo craneoencefálico severo Contusión frontotemporal izquierda (intervenida) Hipertensión intracraneal Neumonía nosocomial
Experiencia en realidad virtual	No, pero presenta experiencia en ejercicios de entrenamiento de la videoconsola Wii.

Equipo utilizado	1º Versión de HTC Vive.

Condiciones de prueba del jugador 1

TABLA 5.3. CONDICIONES DE PRUEBA DEL JUGADOR 1

Parámetros	Simple	Complejo	Complejo
Duración	100s	100s	100s
Velocidad Obstáculos	3u/s	3u/s	2u/s
Intervalo tiempo obstáculos	3s	3s	3s
Total de obstáculos azules	21	20	26
Total de obstáculos rojos	10	10	4
Total de obstáculos	31	30	30

Resultados obtenidos por el jugador 1

TABLA 5.4. RESULTADOS OBTENIDOS POR EL JUGADOR 1

Resultados	Simple	Complejo	Complejo
Azules golpeados	15	16	23
Azules golpeados incorrectamente	10	7	8
Azules no golpeados	6	4	3
Rojos golpeados	0	0	0
Rojos golpeados incorrectamente	0	0	0
Rojos no golpeados	10	10	4
Total golpeados	31	30	30
Total no golpeados	16	14	7
Precisión media azules	19,05%	32,67%	41,37%
Precisión media rojos	0%	0,00%	0,00%
Precisión media total	12,90%	21,78%	35,85%

El alumno trasladó el equipo de realidad virtual personalmente a la casa del usuario para realizar las pruebas. Hay que remarcar en la importancia de la correcta colocación de todos los dispositivos para aumentar la sensación de inmersión virtual. En este caso, las pruebas se realizaron en el comedor del usuario y no se obtuvo la mejor calibración posible de los mandos y visores, por tanto, los resultados obtenidos se han visto influenciados. Por otra parte, el usuario presenta problemas en la parte derecha del cuerpo, esto implica que únicamente probó el videojuego con la mano izquierda.

Además, en algunas ocasiones el usuario esquivaba los obstáculos mediante un comportamiento no esperado por parte del alumno. Intuitivamente si el usuario observaba un obstáculo que debe esquivar, movía hacia abajo el sable/varita para que la punta no impactase con el obstáculo (pero en realidad, el obstáculo no debía impactar con ninguna parte del sable/varita). Entonces, el sistema detectaba que el usuario impactaba contra obstáculos donde el usuario tenía la intención de apartarse. Por esta razón, se ha decidido modificar la región de colisión del sable y de la varita. En resumen, se considera aproximadamente un 30% de la parte superior del sable como región colindante y para la varita únicamente la esfera superior, el resto del objeto no afectará

al entrar en contacto con el obstáculo. Con esto, se consigue esquivar los obstáculos apartando la punta del sable/varita y no el objeto entero.

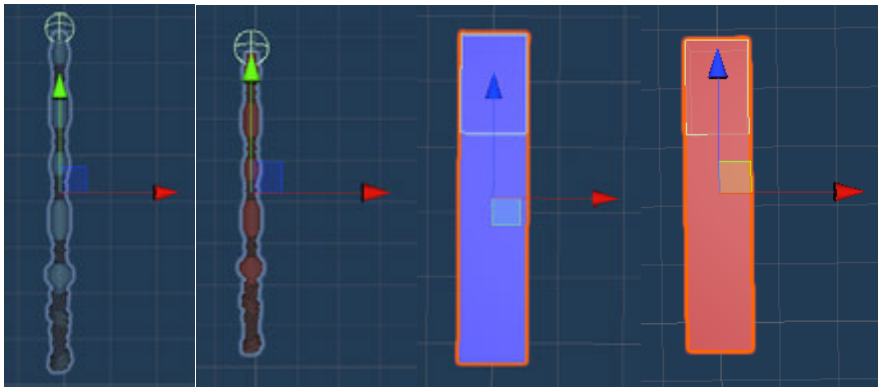


Ilustración 155 - Varitas mágicas con "collider" esférico y sables con "collider" en forma de caja.

También se encontraron errores en el contador de obstáculos rojos destruidos. Cuando un obstáculo rojo era destruido se incrementaba el contador de obstáculos azules destruidos. Por eso, en los resultados se aprecia 0 obstáculos rojos golpeados, pero se vio al paciente cometer errores golpeando obstáculos rojos incorrectamente. Por otra parte, cuando se inicia una prueba, no se conoce la cantidad de obstáculos de cada color porque la generación de los mismos es pseudoaleatoria, entonces es al finalizar la prueba cuando se escribe las condiciones de la prueba (entre otros, la cantidad total de obstáculos de cada color), por consiguiente, el error de actualización del contador también repercute en la información mostrada en las condiciones de la prueba. Por tanto, solamente se han contado los obstáculos rojos que el usuario no ha golpeado, por eso, se observa una diferencia considerable entre la cantidad de obstáculos azules y rojos.

En conclusión, los resultados del usuario no se pueden evaluar debido a la cantidad de factores que han influido negativamente en la experiencia.

La principal ventaja de estas pruebas ha sido la detección de problemas imprevistos en el videojuego y la obtención de opiniones sobre la intuición y facilidad de uso del videojuego. Las dos primeras partidas sirvieron como toma de contacto para el usuario, entonces todavía cometía errores por falta de entendimiento sobre lo que tiene que realizar. No obstante, a partir de la tercera partida, el usuario entendía el funcionamiento del ejercicio y se desenvolvía con mayor naturalidad.

5.4.2. Pruebas conforme el protocolo

5.4.2.1. Jugador 2

Ficha del jugador 2

TABLA 5.5. FICHA DESCRIPTIVA DEL JUGADOR 2

Identificador	**P2**
Sexo	Mujer
Edad	60
Mano dominante	Derecha
Diagnóstico principal	Ninguno

Experiencia en realidad virtual	No
Equipo utilizado	1º Versión de HTC Vive.

Condiciones de prueba del jugador 2

TABLA 5.6. CONDICIONES DE PRUEBA DEL JUGADOR 2

	Prueba 1		Prueba 2	
Parámetros	Simple	Complejo	Simple	Complejo
Duración	100s	100s	60s	60s
Velocidad Obstáculos	3u/s	3u/s	6u/s	6u/s
Intervalo tiempo obstáculos	3s	3s	2s	2s
Total de obstáculos azules	14	16	8	16
Total de obstáculos rojos	17	13	20	10
Total de obstáculos	31	29	28	26

Se puede observar cómo la segunda prueba incrementa la dificultad aumentando la velocidad de los obstáculos y decrementando el intervalo de salida entre cada obstáculo.

Resultados obtenidos por el jugador 2

TABLA 5.7. RESULTADOS OBTENIDOS POR EL JUGADOR 2

	Prueba 1		Prueba 2	
Resultados	Simple	Complejo	Simple	Complejo
Azules golpeados	14	16	8	16
Azules golpeados incorrectamente	1	2	2	6
Azules no golpeados	0	0	0	0
Rojos golpeados	17	13	20	10
Rojos golpeados incorrectamente	2	0	0	3
Rojos no golpeados	0	0	0	0
Total golpeados	31	29	30	26
Total no golpeados	0	0	0	0
Precisión media azules	49,63%	76,24%	60,45%	61,69%
Precisión media rojos	54,26%	87,21%	83,57%	66,07%
Precisión media total	52,17%	81,16%	76,96%	63,38%

El jugador ha realizado todas las pruebas sentado debido al nerviosismo que sentía por probar la realidad virtual, es posible que al sentarse aumente la dificultad por golpear el centro de los obstáculos y se haya visto afectado en los resultados. En ambas pruebas, el jugador ha golpeado todos los obstáculos. Esto demuestra que el jugador mantiene reflejos y capacidad de reacción ante los obstáculos.

En la prueba 1, en el escenario simple el jugador ha golpeado incorrectamente 2 obstáculos rojos y 1 obstáculo azul, esto significa que ha golpeado un 11,76% de los obstáculos rojos incorrectamente y un 7,14% de los obstáculos azules incorrectamente. Sin embargo, obtiene mayor puntuación en los rojos que en los azules, sin embargo, el porcentaje de rojos golpeados incorrectamente es mayor. Esto puede ser porque el jugador ha golpeado con mayor precisión los obstáculos rojos por ser diestro, pero ha golpeado incorrectamente más rojos debido a la falta de experiencia en realidad virtual.

En el escenario complejo, golpea el 100% de los obstáculos rojos, pero no golpea 2 obstáculos azules, es decir no golpea un 12,5% de los obstáculos azules. En este escenario, obtiene una precisión media mayor que en el simple, puede ser porque el escenario simple le ha servido como entrenamiento para este segundo escenario.

En la prueba 2 se ha incrementado la dificultad y el jugador ha mejorado los resultados en el escenario simple y empeorado en el complejo. En el simple, golpea un 25% de los obstáculos azules incorrectamente. Con el transcurso del tiempo, el usuario ha comenzado a fatigarse por el peso de los visores y ha comenzado a jugar de forma más cansada. En consecuencia, es posible que en el escenario complejo haya golpeado un 37,5% de los obstáculos azules incorrectamente. Para esta clase de usuarios sería recomendable reducir el período de las pruebas o realizar una breve pausa.

5.4.4.1 Jugador 3

Ficha del jugador 3

TABLA 5.8. FICHA DESCRIPTIVA DEL JUGADOR 3

Identificador	**P3**
Sexo	Hombre
Edad	25
Mano dominante	Izquierda
Discapacidad	Ninguna
Experiencia en realidad virtual	No
Equipo utilizado	HTC Vive Primera versión

Este jugador tiene la peculiaridad de ser zurdo, no padecer de enfermedades y no haber experimentado con la realidad virtual. Por lo que resultará bastante útil para conocer posibles distinciones del videojuego entre zurdos y diestros y conclusiones sobre la interfaz.

Condiciones de prueba del jugador 3

TABLA 5.9. CONDICIONES DE PRUEBA DEL JUGADOR 3

	Prueba 1		**Prueba 2**	
Parámetros	Simple	Complejo	Simple	Complejo
Duración	100s	100s	60	60
Velocidad Obstáculos	3u/s	3u/s	9u/s	9
Intervalo tiempo obstáculos	3s	3s	1s	1s
Total de obstáculos azules	14	16	26	30
Total de obstáculos rojos	17	13	30	22
Total de obstáculos	31	29	56	52

Se puede observar cómo la segunda prueba tiene la dificultad máxima: menor intervalo posible de tiempo y mayor velocidad permitida de obstáculos.

Resultados obtenidos por el jugador 3

TABLA 5.10. RESULTADOS OBTENIDOS POR EL JUGADOR 3

	Prueba 1		Prueba 2	
Resultados	Simple	Complejo	Simple	Complejo
Azules golpeados	14	16	26	30
Azules golpeados incorrectamente	0	0	0	0
Azules no golpeados	0	0	0	0
Rojos golpeados	17	13	30	22
Rojos golpeados incorrectamente	0	0	0	0
Rojos no golpeados	0	0	0	0
Total golpeados	31	29	92	92
Total no golpeados	0	0	1	0
Precisión media azules	100%	100%	97,48%	99,54%
Precisión media rojos	98,31%	100%	96,83%	99,71%
Precisión media total	99,07%	100%	97,13%	99,61%

El jugador ha obtenido resultados satisfactorios en la prueba 1. No ha golpeado ningún obstáculo de forma incorrecta y tampoco hay obstáculos que se hayan escapado. Para el escenario simple la cantidad de obstáculos rojos es ligeramente superior a los azules, es decir, hay mayor cantidad de obstáculos a golpear con la derecha que con la izquierda. Para el escenario complejo la cantidad de obstáculos azules es ligeramente superior a los rojos, es decir, hay mayor cantidad de obstáculos a golpear con la izquierda que con la derecha. La precisión media obtenida en ambas manos en ambos escenarios supera el 95%, por lo que, se considera que el jugador no ha tenido problemas al ser zurdo y la interfaz le ha parecido intuitiva sin ningún tipo de problema visual.

Debido a la destreza demostrada en la prueba 1, se ha incrementado considerablemente la velocidad de los obstáculos en la prueba 2. En este caso, el jugador ha vuelto a golpear todos los obstáculos con una precisión muy próxima al 100%.

En general, ha obtenido mejores resultados en el escenario complejo, esto puede ser porque primero se realiza la prueba en el escenario simple donde obtiene práctica. Este jugador presenta facilidad de adaptación a los entornos y elevada capacidad de reacción.

5.5. Comparación de los resultados obtenidos por los jugadores

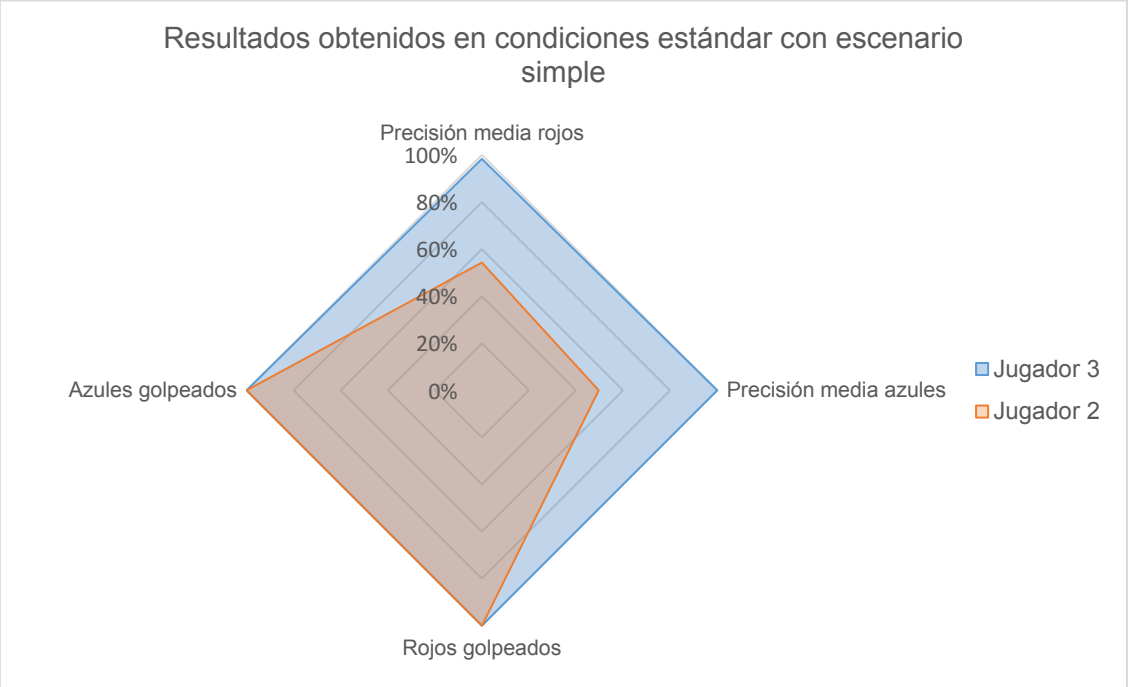
En resumen, ambos jugadores han golpeado todos los obstáculos, pero el jugador 3 ha logrado resultados considerablemente mejores. La comparación se basará en la primera prueba de cada jugador, ya que, utilizan las mismas condiciones.

5.5.1 Comparación de resultados en escenario simple

A continuación, se muestra un resumen de los porcentajes obtenidos en el escenario simple en condiciones estándar por los jugadores.

TABLA 5.11. COMPARATIVA DE RESULTADOS DE JUGADORES EN ESCENARIO SIMPLE

Variables de resultados	Jugador 2	Jugador 3
Azules golpeados	100%	100,00%
Azules golpeados incorrectamente	7,14%	0,00%
Azules no golpeados	0,00%	0,00%
Rojos golpeados	100,00%	100,00%
Rojos golpeados incorrectamente	11,76%	0,00%
Rojos no golpeados	0,00%	0,00%
Total de golpeados	100,00%	100,00%
Total de no golpeados	0,00%	0,00%
Precisión media azules	49,63%	100,00%
Precisión media rojos	54,26%	98,31%
Precisión media total	52,17%	99,07%



Gráfica 3. Gráfica de comparativa de resultados de jugadores en escenario simple

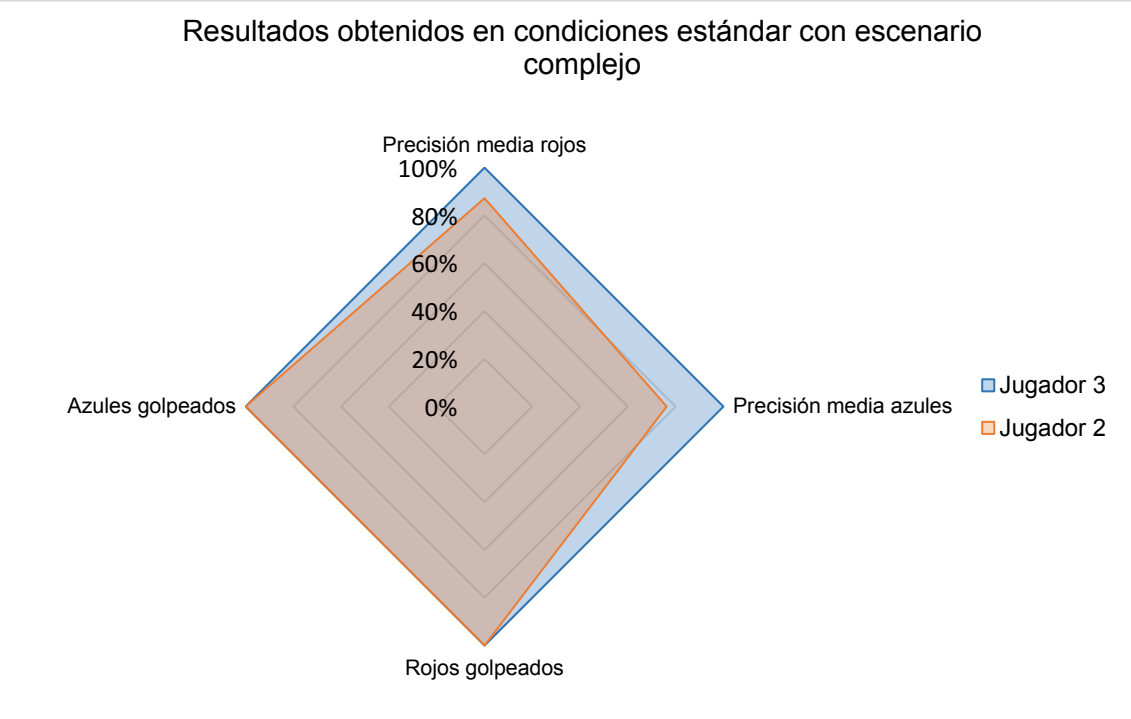
Los datos mostrados en la gráfica son los considerados más relevantes. Se puede observar cómo ambos jugadores han golpeado todos los obstáculos, pero el jugador 3 ha obtenido mayor precisión en los dos colores de obstáculos.

5.5.2 Comparación de resultados en escenario complejo

A continuación, se muestra un resumen de los porcentajes obtenidos en el escenario simple en condiciones estándar por los jugadores.

TABLA 5.12. COMPARATIVA DE RESULTADOS DE JUGADORES EN ESCENARIO SIMPLE

Variables de resultados	Jugador 2	Jugador 3
Azules golpeados	100,00%	100,00%
Azules golpeados incorrectamente	12,50%	0,00%
Azules no golpeados	0,00%	0,00%
Rojos golpeados	100,00%	100,00%
Rojos golpeados incorrectamente	0,00%	0,00%
Rojos no golpeados	0,00%	0,00%
Total de golpeados	100,00%	100,00%
Total de no golpeados	0,00%	0,00%
Precisión media azules	76,24%	100,00%
Precisión media rojos	87,21%	100,00%
Precisión media total	81,16%	100,00%



Gráfica 4. Gráfica de comparativa de resultados de jugadores en escenario complejo

Los datos mostrados en la gráfica son los considerados más relevantes. Se puede observar cómo ambos jugadores han golpeado todos los obstáculos y el jugador 2 obtiene resultados considerablemente mejores que en el escenario simple.

5.5. Conclusiones

El videojuego planteado a los jugadores resulta intuitivo y con controles agradables. Se ha visto que con este ejercicio se puede obtener información de interés acerca de las capacidades del jugador.

Este videojuego resulta conveniente de practicar para mejorar aquellas capacidades reducidas del jugador de una manera amena y entretenida.

6. PLANIFICACIÓN DEL TRABAJO

6.1. Introducción

En esta sección se mostrará la planificación del trabajo realizado. Se mostrará la planificación realizada inicialmente durante el proyecto y la resultante al final del proyecto acompañadas respectivamente del cronograma de actividades y del diagrama de Gantt. Finalmente, se expondrá la comparativa entre cada planificación y se cerrará con las conclusiones.

6.2. Planificación inicial

Se van a mostrar las actividades por las que inicialmente quedó dividido el desarrollo del proyecto y el tiempo estimado para su elaboración.

6.2.1. Cronograma de actividades para la planificación inicial

- 1. Documentación: Presente durante todo el desarrollo del proyecto.
- 2. Aprendizaje y formación del motor gráfico: Consiste en la búsqueda y lectura de manuales para conocer la información primordial del motor gráfico. También se realizarán proyectos para obtener nociones sobre el lenguaje de programación CSharp y estudiar las mecánicas de físicas de los videojuegos, esto incluye las fuerzas gravitatorias de los objetos y las colisiones entre los mismos. También se realizan diferentes pruebas de audio de ambiente.
- 3. Aprendizaje y formación del diseño y modelado 3D: Trata sobre el aprendizaje del programa 3DS MAX gracias a la adquisición de manuales sobre el programa. En esta tarea también se incluye la investigación de diferentes plugins.
- 4. Desarrollo: Trata sobre los proyectos realizados para la obtención del videojuego. Esto incluye tanto el proyecto principal de Unity con los modelos del entorno y la implementación de la funcionalidad como los proyectos requeridos para el diseño y modelado de objetos tridimensionales de 3DS MAX.

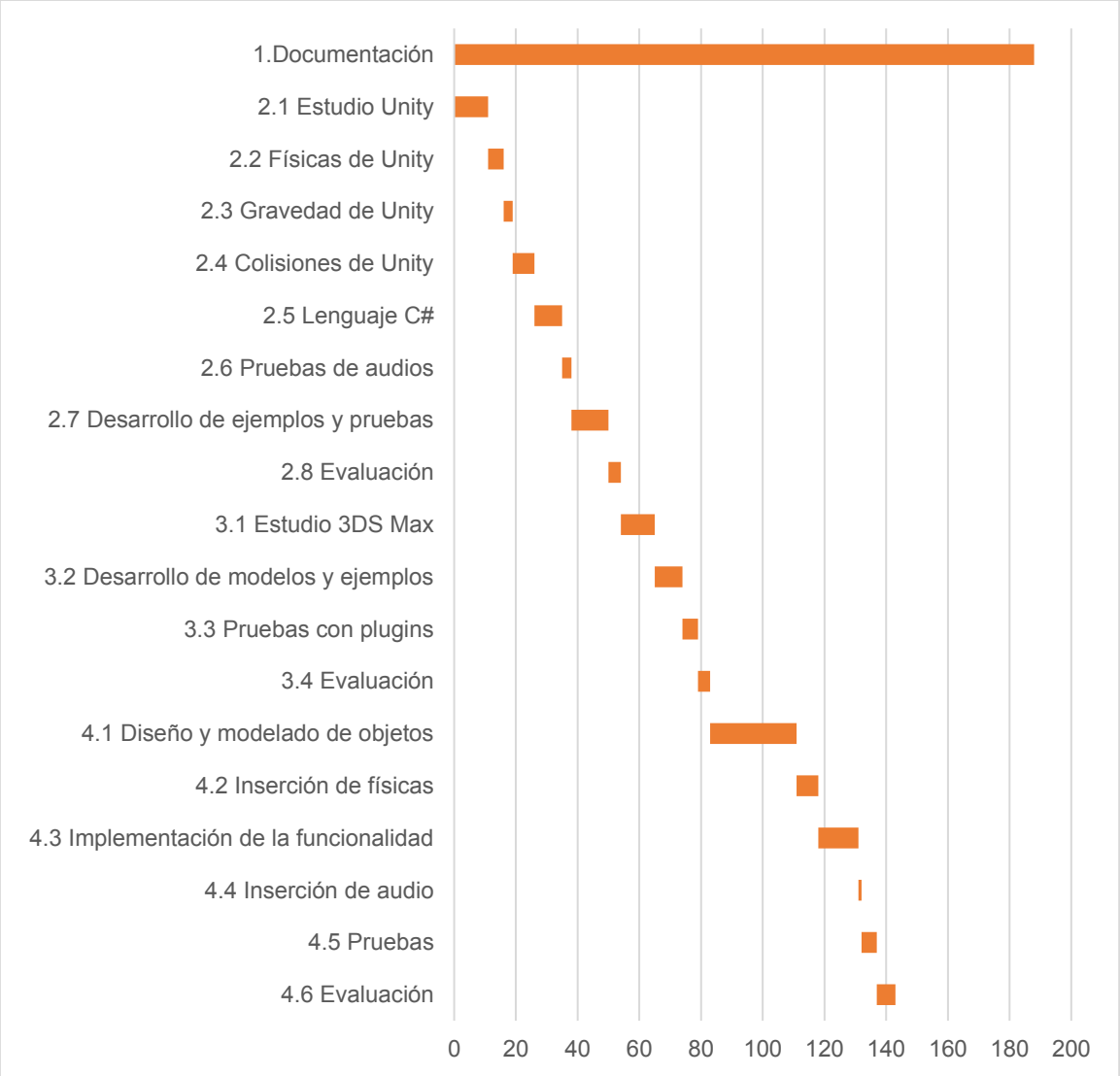
TABLA 6.1. CRONOGRAMA DE ACTIVIDADES PARA LA PLANIFICACIÓN INICIAL

Tarea	Cantidad de Horas
1. Documentación	175
2. Aprendizaje y formación del motor gráfico	48
2.1 Estudio Unity	10
2.2 Físicas de Unity	5
2.3 Gravedad de Unity	3
2.4 Colisiones de Unity	5
2.5 Lenguaje CSharp	8
2.6 Pruebas de audios	3
2.7 Desarrollo de ejemplos y pruebas	10
2.8 Evaluación	4
3. Aprendizaje y formación del diseño y modelado 3D	27
3.1 Estudio 3DS MAX	10
3.2 Desarrollo de modelos y ejemplos	8
3.3 Pruebas con plugins	5
3.4 Evaluación	4

4. Desarrollo	55
4.1 Diseño y modelado de objetos	23
4.2 Inserción de físicas	7
4.3 Implementación de la funcionalidad	13
4.4 Inserción de audio	1
4.5 Pruebas	5
4.6 Evaluación	6
Total	305

Se estima una cantidad total de 305 horas.

6.2.2. Diagrama de Gantt de planificación inicial



Gráfica 5. Diagrama de Gantt de la planificación inicial.

La documentación aparece paralelamente a todas las tareas porque estará presente durante todo el desarrollo del proyecto.

6.3. Planificación final

Se van a mostrar las actividades por las que finalmente quedó dividido el desarrollo del proyecto y el tiempo resultante para su elaboración.

6.3.1. Cronograma de actividades para la planificación final

Aparentemente las actividades principales por las que se ha dividido el desarrollo del proyecto son las mismas que las iniciales.

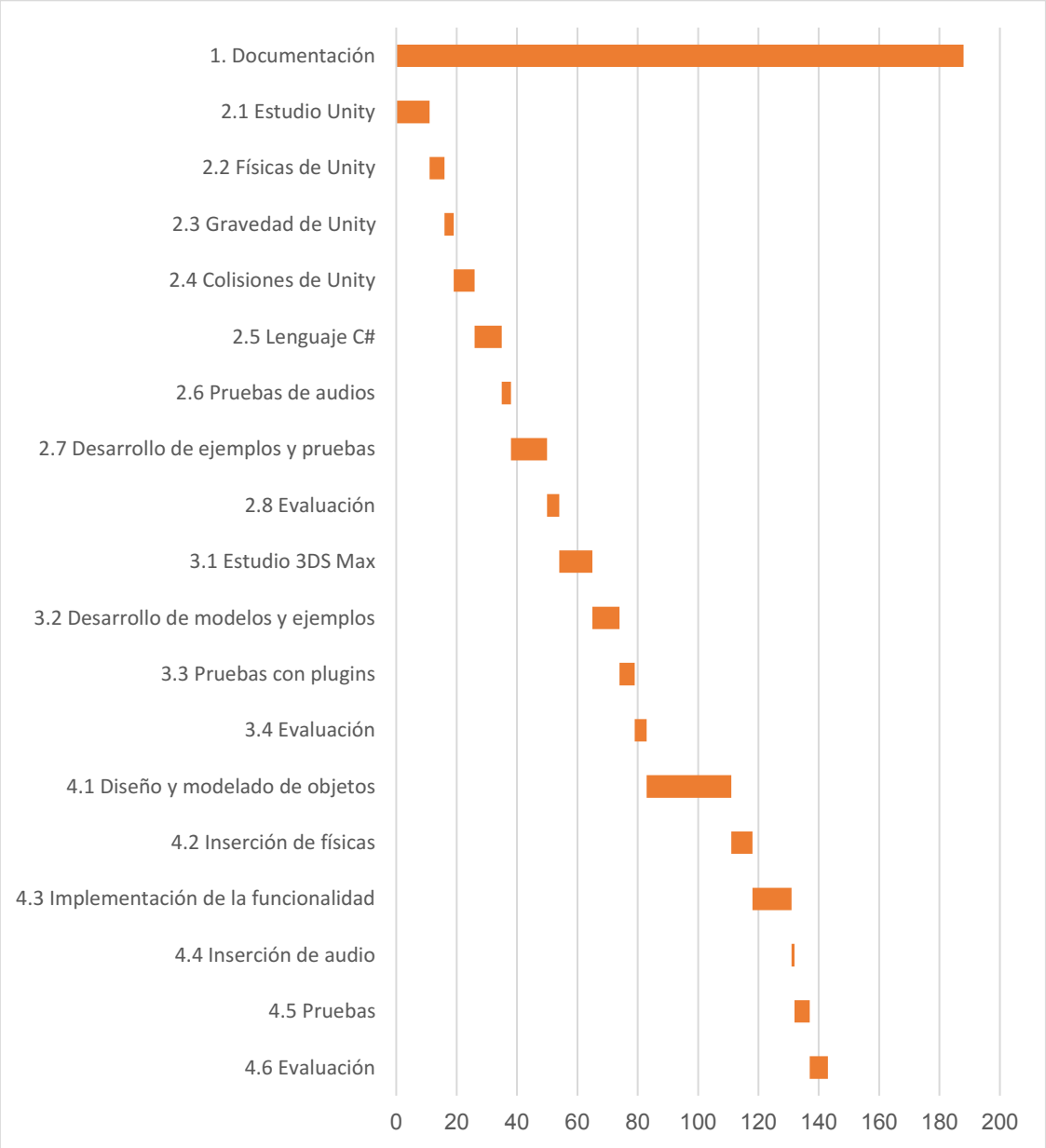
- Aprendizaje y formación del motor gráfico: Se han realizado partes de videojuegos más simples para entender el funcionamiento del lenguaje CSharp y las físicas de los objetos. También se han probado distintos tipos de colisiones y se han alcanzado las conclusiones.
- Aprendizaje y formación del diseño y modelado 3D: Se han elaborado diseños y modelados de objetos más sencillos para contemplar la cantidad de propiedades que existen en cada objeto. También se realizó una selección de las propiedades más interesantes para centrarse en conseguir mejores acabados utilizando las seleccionadas. También se han descargado plugins y se han probado para observar los efectos que consiguen.
- Documentación: Durante los períodos de pruebas con cualquier programa, se ha estado documentando para mantener registrada la información adquirida.
- Desarrollo: Se ha utilizado el conocimiento obtenido en las fases de aprendizaje para diseñar, modelar y programar el funcionamiento del videojuego.

TABLA 6.2. CRONOGRAMA DE ACTIVIDADES PARA LA PLANIFICACIÓN FINAL

Tarea	Cantidad de Horas
1.Documentación	188
2. Aprendizaje y formación del motor gráfico	54
2.1 Estudio Unity	11
2.2 Físicas de Unity	5
2.3 Gravedad de Unity	3
2.4 Colisiones de Unity	7
2.5 Lenguaje CSharp	9
2.6 Pruebas de audios	3
2.7 Desarrollo de ejemplos y pruebas	12
2.8 Evaluación	4
3. Aprendizaje y formación del diseño y modelado 3D	29
3.1 Estudio 3DS MAX	11
3.2 Desarrollo de modelos y ejemplos	9
3.3 Pruebas con plugins	5
3.4 Evaluación	4
4. Desarrollo	60
4.1 Diseño y modelado de objetos	28
4.2 Inserción de físicas	7
4.3 Implementación de la funcionalidad	13
4.4 Inserción de audio	1
4.5 Pruebas	5

4.6 Evaluación	6
Total	331

6.3.2. Diagrama de Gantt de planificación final



Gráfica 6. Diagrama de Gantt para la planificación final.

La documentación aparece paralelamente a todas las tareas porque estará presente durante todo el desarrollo del proyecto.

6.4. Comparativa de la cantidad de horas estimadas frente a las realizadas

A continuación, se va a mostrar una tabla comparativa con las horas estimadas y resultantes en cada fase del proyecto. También se mostrará la diferencia definida como las horas resultantes menos las horas estimadas.

TABLA 6.3. COMPARATIVA DE LA CANTIDAD DE HORAS ESTIMADAS FRENTE A LAS REALIZADAS

Tarea	Horas estimadas	Horas resultantes	Diferencia
1. Documentación	175	188	13
2. Aprendizaje y formación del motor gráfico	48	54	6
2.1 Estudio Unity	10	11	1
2.2 Físicas de Unity	5	5	0
2.3 Gravedad de Unity	3	3	0
2.4 Colisiones de Unity	5	7	2
2.5 Lenguaje CSharp	8	9	1
2.6 Pruebas de audios	3	3	0
2.7 Desarrollo de ejemplos y pruebas	10	12	2
2.8 Evaluación	4	4	0
3. Aprendizaje y formación del diseño y modelado 3D	27	29	2
3.1 Estudio 3DS MAX	10	11	1
3.2 Desarrollo de modelos y ejemplos	8	9	1
3.3 Pruebas con plugins	5	5	0
3.4 Evaluación	4	4	0
4. Desarrollo	55	60	5
4.1 Diseño y modelado de objetos	23	28	5
4.2 Inserción de físicas	7	7	0
4.3 Implementación de la funcionalidad	13	13	0
4.4 Inserción de audio	1	1	0
4.5 Pruebas	5	5	0
4.6 Evaluación	6	6	0
Total	305	331	39

La estimación de la duración del proyecto frente a la duración real del proyecto mantiene una diferencia de 39 horas. Esto es porque inicialmente se pensaba realizar menor cantidad de diseños, modelos y pruebas.

En concreto para la fase de aprendizaje y formación del motor gráfico hay una diferencia de 6 horas, este incremento se observa principalmente por la cantidad de horas en el desarrollo de ejemplos y pruebas. También, el estudio de las posibilidades y limitaciones que ofrece Unity junto con el estudio de las colisiones han supuesto mayor cantidad de tiempo.

Para la fase de documentación, se han tardado 13 horas adicionales.

Para el aprendizaje y formación del diseño y modelado 3D se han tardado 2 horas extras porque se desconocía de la elevada cantidad de propiedades que contienen los objetos. Gran parte del tiempo se ha llevado en la prueba de las propiedades y el entendimiento de las consideradas más útiles para el diseño concreto.

En la fase del desarrollo del proyecto hay una diferencia de 5 horas. Esto se debe principalmente al modelado de elementos extras que no se pensaron inicialmente como los árboles, rocas, flores y setas.

6.5. Conclusiones

Realizar una planificación del trabajo es la tarea principal para empezar a elaborar un proyecto, y con más razón, para los casos con escasa experiencia donde el alumno debe aprender a utilizar varios programas totalmente nuevos para él mismo. Es importante organizarse las diferentes fases del proyecto, ya que, esto ha permitido facilitar la toma de decisiones para estudiar los programas y poder avanzar durante el desarrollo del proyecto.

7. ASPECTOS ECONÓMICOS Y LEGALES

7.1. Introducción

El propósito de esta sección es exponer la implicación del sector de los videojuegos en la economía de España y el presupuesto que conlleva desarrollar este proyecto. También se expondrán las clasificaciones que adopta un videojuego se finalizará con las conclusiones.

7.2. Entorno socioeconómico

7.2.1. En el sector

La información que se va a presentar en este apartado se ha recogido del anuario de la Asociación Española de Videojuegos (AEVI) de 2018 [114].

En el año 2018, los videojuegos eran una opción de entretenimiento muy elegida por los españoles, había 16,8 millones de jugadores de todas las edades. Lo que se traduce en que el sector del videojuego iba a generar millones de ingresos. En concreto, España obtuvo un total de ingresos de 1.530 millones de euros por los videojuegos, de los cuales 680 millones eran por el área online.

La plataforma más utilizada para jugar por los españoles son las videoconsolas, con un 26% del total de los jugadores. Sin embargo, el ordenador y los smarthphones, que están en continuo crecimiento, se encuentran con un 21%. Se espera que con el paso de los años estos dos dispositivos adelanten a las videoconsolas, ya que, el ordenador y los móviles sirven tanto como herramienta del trabajo como de entretenimiento. Las consolas portátiles y las tabletas mantienen menor porcentaje con 10% y 8% respectivamente, debido a que los móviles también son portátiles y tienen una gran variedad de videojuegos, lo que les convierte en una principal competencia contra el mercado de entretenimiento portátil.

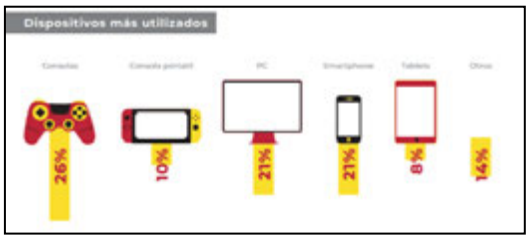


Fig.156 – Dispositivos más utilizados en España [114].

Si se estudia el porcentaje de uso de cada dispositivo en función de la edad del usuario, se puede observar que es en el rango de edad de 11 a 14 años cuando más se utiliza las videoconsolas. Y es que los niños al no tener que salir a trabajar permanecen más tiempo en casa, por lo que, disponen de más tiempo y mayor comodidad para utilizar dispositivos como consolas conectadas a televisor.

A partir de los 15 años, las personas empiezan a utilizar más las redes sociales y hacen mayor uso de internet, lo que los lleva a reducir el uso de las consolas, por lo que, se equipara el uso entre móvil y consola.

Sobre las edades 25-44, la mayoría de las personas tienen trabajo fuera de casa. Por tanto, tienen menos

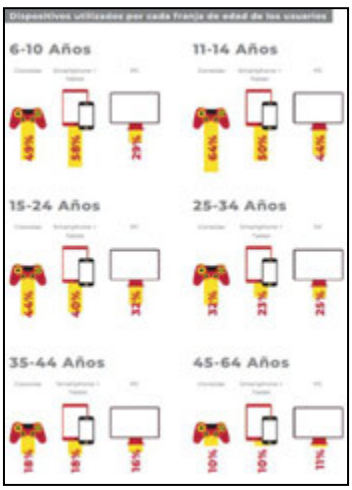


Fig.157 – Dispositivos más utilizados por edades en España [114].

tiempo para jugar a las videoconsolas y el entretenimiento lo buscan a lo largo del día mediante un uso ligero del móvil. Y a partir de 45 años, se pierde mayor interés por los videojuegos.

Es tal el interés por los videojuegos que se han creado competiciones oficiales internaciones de videojuegos. Estas competiciones se han convertido en un deporte electrónico conocido como e-sport, que obtuvieron casi 7 millones de espectadores en España.

La popularidad de los e-Sport está aumentando considerablemente gracias empresas de telecomunicaciones. Desde 2017, Movistar patrocinaba Electronic Sports League (ESL) y Mediapro compraba Liga de Videojuegos Profesional. Gracias a todos estos acontecimientos, el sector de los videojuegos ha logrado más ingresos que la industria del cine (585,7 millones de euros, según Comscore) y de la música (237,2 millones de euros, según Promusicae).

Los E-Sports han llegado a superar los premios más elevados de alguno de los deportes más populares. Con el mundial del videojuego de Fortnite el mayor premio era de 2.672.000€, cifras que superan otros campeonatos deportivos como el Roland Garros de tenis con 2.200.000€, Masters de Augusta de Golf con 1.763.000€ o el mundial de dardos con 570.000€.

En España, el sector de los videojuegos genera cerca de 9.000 empleos directos y de cada uno surgen 2,6 empleos en otros sectores. Cada vez se están aumentando los requisitos mínimos en las ofertas de empleo para especializarse en distintos ámbitos de un videojuego: diseñador gráfico, modelador de entornos 3D, lenguajes de programación, tratamiento de datos… Por lo que se espera que en años posteriores este sector genere todavía más ingresos.

7.2.2. Presupuesto del proyecto

A nivel económico no ha resultado excesivamente costoso realizar el videojuego, ya que, el alumno ya disponía de la mayoría de los elementos fundamentales necesarios para el trabajo. Entonces el coste de realizar el videojuego principalmente vendrá por la ambición que se tiene por el proyecto y por la cantidad de horas trabajadas para conseguir las metas buscadas.

Para obtener una estimación de valor numérico de presupuesto, se va a dividir el coste por diferentes tareas:

7.2.2.1. Material

7.2.2.1.1. Portátil

El alumno dispone de un portátil ASUS con 8GB de memoria RAM, una tarjeta gráfica NVIDIA Geforce GTX 950 M y un procesador Intel Core i7-6700HQ.

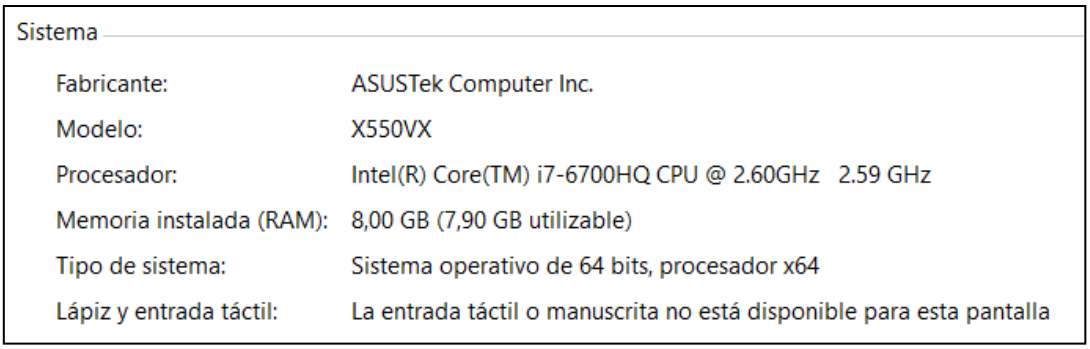


Fig.158 – Propiedades del ordenador del alumno.

El portátil del alumno se ha descatalogado actualmente, por tanto, el precio del portátil se basará en otro muy similar de 979,90€ según la página de ventas online K*riplus* [115].

Teóricamente el tiempo de amortización de un portátil es de cinco años sometido a 8 horas diarias (solamente entre semana), lo que se traduce en 8 horas durante 22 días los 12 meses de los 5 años:

$$\text{Horas de amortización de un portátil} = 8\text{x}22\text{x}12\text{x}5 = 10.560 \text{ horas}$$

Sabiendo que el proyecto ha mantenido una duración de 331 horas, el precio del portátil vendrá dado como:

$$\text{Precio del portátil en el proyecto} = \frac{331\text{h x } 979{,}90€}{10.560\text{h}} = 30{,}71€$$

7.2.2.1.2. Visores de realidad virtual

Los visores de realidad virtual que se van a utilizar se corresponden con la primera versión de las HTC VIVE que fueron lanzadas en 2016.

El precio de las HTC Vive es de 715,00€ según la página de ventas Amazon [116].

El tiempo de amortización de los visores de realidad virtual mantiene mayor dificultad para ser estimado. Ya que, los visores están pensados para una compra única, sin embargo, las empresas “Valve” y “HTC” pueden lanzar nuevas versiones de visores mejores, por consiguiente, se considerará un tiempo de amortización de 5 años al igual que los portátiles.

$$\text{Precio de los visores en el proyecto} = \frac{331\text{h x } 715{,}00€}{10.560\text{h}} = 22{,}41€$$

7.2.2.1.3. Resumen de los gastos de material

TABLA 7.1. RESUMEN DE LOS GASTOS DE MATERIAL

Material	Coste unidad	Coste en el proyecto
Portátil Asus	979,90€	30,71€
HTC Vive	715,00€	22,41€
Total	--	53,12€

Los costes de material requieren un total de 53,12 €.

7.2.2.2. Licencias

7.2.2.2.1. Blackboard Collaborate

La mayoría de las reuniones del alumno con el tutor se han realizado mediante llamadas por internet. Estas llamadas se han realizado a través de la plataforma Blackboard Collaborate, un software que permite la comunicación entre dos o más usuarios por escritura, voz y videollamada. Según la página bit4learn como la universidad tiene más de 300 alumnos matriculados, el precio por alumno es de 1.200$ al año [117]. En abril de 2020, un dólar equivale a 0,93€, por tanto, el precio por año sería de 1.116€.

Debido a que la plataforma se ha utilizado únicamente para el desarrollo del proyecto y se desconoce si se utilizará en el futuro, no se puede considerar que haya una reutilización de la cuenta de usuario. Por tanto, el precio de la cuenta al año es el precio requerido para el desarrollo del TFG.

7.2.2.2.2. Windows 10 Pro

El sistema operativo base utilizado durante todo el proyecto ha sido Windows 10 Pro. Según la página oficial de Windows, el precio es de 259,00€ [118]. En un principio se espera utilizar este sistema operativo durante los años de vida del portátil, por tanto, la amortización de este sistema operativo será la misma del ordenador (5 años).

$$\text{Coste licencia Windows en el proyecto} = \frac{331\text{h x } 259{,}00€}{10.560\text{h}} = 8{,}12€$$

7.2.2.2.3. Office 365 Empresa Premium

A lo largo del desarrollo del proyecto se han utilizado diferentes aplicaciones pertenecientes a Microsoft Office, en concreto: Microsoft Word, Microsoft Excel, Microsoft PowerPoint y Microsoft Outlook. Para ello, se ha hecho uso de la versión Office 365 Empresa Premium con un precio de 10,50 €/mes [119].

Suponiendo que todas las semanas se han trabajado 20 horas, las 331 horas del proyecto se corresponden con 17 semanas, lo que se traduce en la suscripción de la licencia durante 5 meses. Por tanto, el coste de la contratación de Office 365 Empresa Premium será de:

$$5 \text{ meses} * 10{,}50 €/\text{meses} = 52{,}50€$$

7.2.2.2.4. Resumen de los gastos de licencias

TABLA 7.2. RESUMEN DE LOS GASTOS DE LICENCIAS

Material	Coste en el proyecto
Blackboard Collaborate	1.116€
Windows 10 Pro	8,12€
Office 365 Empresa Premium	52,50€
Total	1.176,62€

Los costes de licencias requieren un total de 1.176,62€.

7.2.2.3. Electricidad

7.2.2.3.1. Consumo de ordenador

Se calculará el gasto por el consumo de luz por el uso del ordenador durante todas las fases del proyecto. Se ha investigado sobre el consumo por hora del ordenador del que dispone el alumno y es de cuatro céntimos por hora [120].

Sabiendo que la cantidad de horas resultantes del proyecto han sido de 331, el gasto eléctrico en consumo del ordenador es de:

$$331 \text{ h x } 0{,}04 \text{ €/h} = 13{,}24 \text{ €}$$

7.2.2.3.2. Iluminación del entorno

Hay que tener en cuenta la iluminación de la habitación donde se realizaba el trabajo. Para ello, se ha hecho uso de una lámpara con una bombilla de 60 vatios. Iberdrola es una empresa dedicada a la producción, distribución y comercialización de energía que ofrece diferentes tarifas por hora.

El alumno cuenta con el "Plan Estable" de Iberdrola, entonces el gato por hora por cada kilovatio es de 0.1349€/kWh [121].

Sin embargo, el alumno no ha hecho uso de la lámpara durante las 331 horas de trabajo, por lo que se va a considerar que la ha utilizado durante la mitad de ese tiempo.

El gasto eléctrico en iluminación es de:

$$\left(\frac{331}{2}\right) \text{h x } 0{,}06\text{kW x } 0{,}1349 \text{ €/kWh} = 1{,}34\text{€}$$

7.2.2.3.3. Resumen de los gastos de electricidad

TABLA 7.3. RESUMEN DE LOS GASTOS DE ELECTRICIDAD

Material	Coste en el proyecto
Consumo de ordenador	13,24€
Iluminación del entorno	1,34€
Total	14,58€

Los gastos de electricidad requieren un total de 14,58 €.

7.2.2.4. Internet

El proyecto se ha realizado online con conexión a internet, esto incluye, entre otros, la descarga de Unity, 3DS MAX, la búsqueda de ayudas por foros, visualizar videotutoriales, contactar mediante llamadas y correo electrónico con el tutor y buscar documentación y manuales.

Para ello, se ha hecho uso de una conexión de fibra óptica de 100MB de bajada y 50 MB de subida. El precio de dicha conexión es de 35€/mes. Anteriormente se vio que la duración del proyecto en 20 horas semanales se corresponde con 5 meses. Por tanto, el coste de la contratación de internet será de:

$$5 \text{ meses x } 35{,}00 \text{ €/meses} = 175{,}00\text{€}$$

Los gastos de internet requieren un total de 175,00 €.

7.2.2.5. Contratación del personal

Principalmente en este proyecto ha hecho falta un analista, un gestor de calidad, un desarrollador y un gestor de pruebas. Puesto que el alumno ha desempeñado todos los roles, el coste por horas se asignará como un valor que represente el coste en conjunto. Es por ello, por lo que el coste del personal es de 30€/h.

Basándose en la duración del proyecto de 331 horas, el coste total de contratación del personal es de:

$$331\text{h} \times 30\ €/\text{h} = 9.930{,}00€$$

7.2.2.6. Resumen del presupuesto

Se va a mostrar una tabla resumen del presupuesto requerido para este proyecto:

TABLA 7.4. RESULTADO DEL PRESUPUESTO DEL PROYECTO ELABORADO

Tarea	Coste
Material	53,12€
Licencias	1.176,62€
Electricidad	14,58€
Internet	175,00€
Contratación del personal	9.930,00€
Total (sin IVA)	11.349,32€
Contingencias (10%)	1.134,93 €
Total con contingencias (sin IVA)	12.484,25 €
IVA (21%)	2.621,69 €
Total con contingencias (con IVA)	15.105,94 €

Se puede observar que, a nivel económico, el coste es bastante bajo debido a que el alumno está aprovechando todos los recursos de los que ya dispone.

7.3. Restricción legal

Existe gran variedad de videojuegos, cada uno con una temática y forma de representar el contenido diferente. Es posible que los niños jueguen a videojuegos que muestren demasiado contenido explícito violento y lleguen a pensar que el comportamiento del videojuego es el normal en la vida real. Con el objetivo de este problema, se ha creado en la Unión Europea el sistema Pan European Game Information (PEGI) [15].

PEGI se encarga de clasificar un videojuego en función de la edad mínima que debería tener el jugador y en función de la descripción del contenido del videojuego.

Clasificación por edad [15].

Fig.159 – Etiquetas PEGUI por edades [15].

- PEGI 3: representa que el juego es apropiado para todos los públicos. El juego no contiene personajes que se puedan distinguir en el mundo real, para conseguir esto, suelen ser videojuegos con estilo de dibujos animados.
- PEGI 7: representa que el juego continúa sin tener restricción de edad, sin embargo, tiene contenido que puede alarmar negativamente al jugador, ya sea por imágenes o sonidos.
- PEGI 12: representa que el juego contiene escenas semi-violentas y/o semi-desnudos. Se tratan de mundos de fantasía, por lo que, carecen de un contexto de mundo real.
- PEGI 16: representa que el juego contiene escenas semi-violentas y/o semi-desnudos. Empiezan a primar las acciones criminales como las drogas y se basan en representaciones de un mundo real.
- PEGI 18: representa que el juego contiene escenas violentas y/o desnudos. Las acciones son muy explícitas.
- PEGI OK: marca que un juego online o un portal web es apto para todos los públicos.

Clasificación por descripción de contenido [15].

Fig.160 – Etiquetas PEGUI por descripción de contenido [15].

- Violencia: representa que el juego contiene escenas de peleas con o sin armas, sangre o heridas graves en los personajes.
- Lenguaje Soez: representa que el juego contiene diálogos entre los personajes con lenguaje vulgar.
- Discriminación: representa que el juego contiene escenas de marginación por género, raza, edad, grupo étnico…
- Drogas: representa que el juego contiene escenas de narcóticos u otros alucinógenos ilegales.
- Miedo: representa que el juego tiene una temática que puede causar miedo al jugador y puede contener escenas perturbadoras.
- Juego: representa que el juego contiene escenas de juegos de azar o apuestas.
- En línea: representa que el juego tiene modo multijugador haciendo uso de internet.
- Sexo: representa que el juego contiene escenas con personajes sin ropa y referencias sexuales.

7.4. Conclusiones

Se ha podido observar que el sector de los videojuegos se encuentra en una época de constante crecimiento y que no parece que vaya a entrar en desaparición (si llegase a entrar) hasta dentro de muchos años.

8. CONCLUSIONES GENERALES

8.1. Introducción

El propósito de este capítulo es recoger las conclusiones alcanzadas durante el desarrollo del proyecto. Se describirá con detalle los objetivos cumplidos, los problemas encontrados y las futuras líneas de trabajo del prototipo de videojuego desarrollado en el marco de este proyecto.

8.2. Cumplimiento de objetivos

El proyecto comenzó con el objetivo de desarrollar un prototipo de realidad virtual que simule un ejercicio de rehabilitación para pacientes con trastornos neurológicos. Principalmente, el videojuego consiste en la destrucción de una secuencia de obstáculos que se irán mostrando al paciente. El paciente deberá tocar cada obstáculo de la manera más rápida y precisa posible.

En cuanto a los requisitos funcionales y no funcionales, se ha visto la necesidad del obligado cumplimiento de todos los requisitos exceptuando los requisitos del sonido y la música. Esto es porque se considera que el jugador no necesita de música para realizar el ejercicio.

Para la creación del videojuego, el alumno ha partido con escasa experiencia en los programas utilizados:

3DS MAX: Abarca el modelado de todos los objetos 3D que se muestran en el escenario del videojuego. A pesar de existir librerías y modelos de uso libre y gratuito, el alumno ha decidido diseñar y modelar todos los objetos tridimensionales. También se ha probado la instalación de plugins en el programa únicamente con el fin de aprender a utilizarlos, sin embargo, no se ha hecho uso de ningún plugin en sus modelos finales para conseguir la mayor instrucción posible sobre este programa. Los objetos modelados y texturizados con este programa serán exportados y posteriormente importados al programa Unity.

Unity: abarca la creación del escenario del videojuego y su funcionamiento. Este programa ha permitido la creación e importación de los objetos de la escena, inserción de música de fondo, texturización de diferentes elementos, posición de todos los objetos tridimensionales, animación de objetos y programación de funcionamiento de algunos de ellos.

El videojuego resultante se ha analizado con un sistema de evaluación que divide al sistema en criterios independientes para aumentar el grado de precisión de evaluación. El resultado es que el videojuego cumple con los objetivos deseados y en algunos casos ha mejorado el resultado deseado.

También, el videojuego se ha probado con personas reales. El videojuego ha resultado ser intuitivo y fácil de usar para las personas probadas. De las pruebas realizadas hasta el momento, se puede observar que este juego presenta un gran potencial para medir y obtener información sobre el estado físico de los jugadores.

A nivel de crecimiento personal, el desarrollo de este videojuego ha permitido cumplir con el objetivo de aprender a utilizar un motor gráfico, así como de aprender

sobre el diseño y modelado de objetos tridimensionales. Todo ello, aprendiendo de manera autónoma.

8.3. Problemas encontrados

A lo largo del desarrollo del proyecto se han encontrado diferentes problemas:

- Diseño del videojuego: inicialmente no se tenía en mente una estructura clara y sólida sobre la forma que adoptaría el videojuego. Únicamente se tenía clara la idea de destruir obstáculos lo más al centro posible, entonces resultó complicado establecer las primeras ideas de diseño. A raíz de las primeras ideas de diseño, se pudieron realizar mejoras que se fueron añadiendo de forma iterativa.
- Diseño de los objetos tridimensionales: crear una imagen mental del objeto que se pretende diseñar es tarea complicada. El diseño de los objetos ha pasado por diferentes fases y prototipos para elegir la forma y las posibles deformaciones que debería tener. Todos los diseños primero tomaron forma en papel para saber qué funcionalidad buscar y aprender en 3DS MAX. Esta metodología es muy importante porque si se parte sin una imagen mental del diseño y se empieza a buscar opciones directamente en 3DS MAX, puede resultar abrumador y estresante, ya que, al tener tantas opciones resultará complicada elegir cuál utilizar para hacer el modelado.
- Físicas de Unity: inicialmente no se estaba utilizando correctamente los “collider” en los obstáculos. Primero se optó en crear “collider” esféricos que englobasen al obstáculo en sí mismo. Sin embargo, esto hacía que el jugador destruyera el obstáculo demasiado pronto y se dificultaba obtener buenos resultados. También se probó a utilizar “collider” bidimensionales, de esta forma, se podía crear un “collider” circular que englobase al obstáculo y se destruyera cuando realmente corresponde. No obstante, los “collider” bidimensionales tienen problemáticas en Unity porque internamente no funcional de la misma forma que los tridimensionales, entonces nunca detectaban la colisión y actuaban como obstáculos indestructibles al usuario.
- Cambios de escena: cuando el jugador finalizaba la partida en cualquier escenario (simple o complejo) y regresaba automáticamente al menú principal, aparecía con 4 manos en vez de 2. Este problema se daba porque inicialmente en el menú principal se instanciaba un jugador con 2 manos y al cambiar de escenario, esa instancia no se eliminaba de memoria. Entonces, al regresar al menú principal existían 2 instancias del jugador: la instancia inicial antes de cambiar de escena y la nueva instancia creada al entrar en el menú principal.
- Puntuación de golpe: inicialmente el jugador obtenía una puntuación por golpe del obstáculo, en función de la distancia al centro (en coordenadas bidimensionales) del obstáculo. El problema de esta implementación es que el jugador prácticamente nunca obtendría 100 puntos porque es muy poco probable que golpee exactamente al punto de las coordenadas bidimensionales del centro. Entonces, la puntuación obtenida se encontraba en media incluso por debajo del 80% (en casos aparentemente de buenos golpes). Es por ello, por lo que se creó posteriormente un rango central donde el jugador siempre obtendría 100 puntos si golpea dentro de ese rango.
- Funcionalidades de 3DS MAX: dentro de las funcionalidades utilizadas en el programa 3DS MAX, las más problemáticas han sido la de doblar un objeto, la de retorcer un objeto y la de convertir un objeto a “Editable Poly”. La ventaja de “Editable Poly” es que puedes manipular un objeto por subniveles o capas y también puedes manipular las posiciones de los segmentos del objeto. Los segmentos actúan como vértices de un poliedro para

darle la forma deseada. Sin embargo, aprender a utilizar esto no fue sencillo, ya que no se entendía en un principio cómo manejar las capas que estructuraban un objeto y tampoco cómo agregar nuevas capas sobre ellas.

8.4. Puntos débiles

El videojuego presenta algunas debilidades:

- Destrucción de obstáculos: para destruir un obstáculo el jugador debe evitar el contacto de la parte superior del sable/varita con el obstáculo. No obstante, se ha visto el caso de usuarios que intuitivamente bajan el sable/varita para que desaparezca de su punto visual y considerar que así se está esquivando el obstáculo.
- Finalizar inmediatamente una prueba: si el usuario se cansa de una prueba o ha iniciado una prueba con los parámetros equivocados, el sistema no da la posibilidad de regresar al menú principal durante la prueba, por tanto, tiene que esperar a que termine la prueba.
- Altura del jugador: el videojuego desarrollado no tiene en cuenta la altura del jugador. Sería recomendable crear diferentes versiones del videojuego ajustándose a la altura del mismo.
- Actualización de parámetros del menú principal: cuando el usuario regresa al menú principal después de una partida, los valores de los parámetros vuelven a ser los de defecto. Es posible que intuitivamente el usuario piense que en el menú principal se muestran los parámetros de la última partida que jugó.
- Centro del escenario: El escenario complejo utiliza un camino rocoso para indicar el camino por el que circularán los obstáculos y el escenario simple utiliza una línea blanca. Es posible que se tenga que modificar los caminos para que representen el trayecto de los obstáculos con mayor precisión. Además, el centro del menú principal no coincide exactamente con el centro de los escenarios de prueba, por lo que también deberían de ajustarse entre sí.

8.5. Líneas de futuro

Actualmente el videojuego se encuentra en una versión estable y funcional. Si se arreglasen los puntos débiles, es posible que esta versión se pudiera utilizar en centros de salud u hospitales para medir capacidades de los pacientes.

A nivel de futuro, este videojuego puede mejorarse para medir más características. Se podría introducir un parámetro que indique el tamaño de los obstáculos o dos parámetros para establecer un rango aleatorio de tamaño de obstáculos. También se podría aumentar la cantidad de posibles puntos de aparición de los obstáculos. Se podría incluir “obstáculos trampa”, los cuales tiene que esquivar el paciente en vez de golpear.

En definitiva, este videojuego se puede continuar trabajando para ampliar el análisis de resultados obtenidos por un paciente.

8.6. Conclusiones

Realizar este proyecto ha resultado ser didáctico para el alumno. El alumno ha aprendido sobre los programas utilizados para el desarrollo del videojuego y de la realidad virtual.

REFERENCIAS BIBLIOGRÁFICAS

[1] J. Pedreira, «Conec,» 14 de Febrero 2018. [En línea]. Available: http://www.conec.es/historia/%E2%80%A8el-primer-ordinador-del-mon-compleix-anys/. [Último acceso: 29 de Abril 2020].

[2] M. Montero, «Xombit,» Cultura y Sociedad, 12 de Septiembre 2012. [En línea]. Available: https://xombit.com/2012/09/memex-invento-cambiar-mundo. [Último acceso: 29 de Abril 2020].

[3] P. Delgado, «ABC Blogs,» 04 de Julio 2017. [En línea]. Available: https://abcblogs.abc.es/fahrenheit-451/diseno/el-primer-editor-grafico-interactivo-es-de-1963.html?ref=https%3A%2F%2Fwww.google.com%2F. [Último acceso: 29 de Abril 2020].

[4] C. Hope, «Computer Hope,» 29 de Diciembre 2017. [En línea]. Available: https://www.computerhope.com/issues/ch001083.htm. [Último acceso: 29 de Abril 2020].

[5] Wikipedia, «Wikipedia,» 22 de Octubre 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Ivan_Sutherland. [Último acceso: 29 de Abril 2020].

[6] K. D. Q. Villalobo, «SlideShare,» Universidad Popular del César, 25 02 2016. [En línea]. Available: https://es.slideshare.net/Kanana1993/historia-de-la-computacin-grfica-karen-quiroga. [Último acceso: 29 de Abril 2020].

[7] Wikipedia, «Wikipedia,» 25 de Marzo 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Historia_de_la_animación_por_computadora. [Último acceso: 29 de Abril 2020].

[8] N. F. B. o. Canada, «NFC.CA,» 1973. [En línea]. Available: https://www.nfb.ca/film/animation_from_cape_dorset/. [Último acceso: 29 de Abril 2020].

[9] Wikimedia, «Wikimedia,» 12 de Noviembre 2017. [En línea]. Available: https://commons.wikimedia.org/wiki/File:Shading_models.png. [Último acceso: 29 de Abril 2020].

[10] Wikipedia, «Wikipedia,» 25 de Marzo 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Historia_de_la_animación_por_computadora. [Último acceso: 29 de Abril 2020].

[11] M. J. Clayton, «Papers.Cumincad,» [En línea]. Available: http://papers.cumincad.org/data/works/att/sigradi2005_103.content.pdf. [Último acceso: 29 de Abril 2020].

[12] D. Barnard, «VirtualSpeech,» 06 de Agosto 2019. [En línea]. Available: https://virtualspeech.com/blog/history-of-vr. [Último acceso: 29 de Abril 2020].

[13] A. P. Masa, «Alpoma,» Obsolescencia, 06 de Junio 2009. [En línea]. Available: https://alpoma.net/tecob/?p=1141. [Último acceso: 29 de Abril 2020].

[14] P. Ihnen, «One-Tech,» One-Tech, 21 de Marzo 2018. [En línea]. Available: https://one-tech.es/2018/03/21/historia-de-la-realidad-virtual-parte-1/. [Último acceso: 29 de Abril 2020].

[15] S. Ota, «Sutori,» 2012. [En línea]. Available: https://www.sutori.com/item/1961-headsight-developed-by-comeau-and-bryan-the-headsight-projected-screen-f. [Último acceso: 29 de Abril 2020].

[16] L. Pardo, «Neoteo,» [En línea]. Available: https://www.neoteo.com/el-primer-casco-de-realidad-virtual/. [Último acceso: 29 de Abril 2020].

[17] Teknoplof, «Teknoplof,» [En línea]. Available: https://www.teknoplof.com/2014/01/20/historia-de-la-realidad-virtual-en-el-mundo-de-los-videojuegos/. [Último acceso: 29 de Abril 2020].

[18] Amazon, «Amazon,» 12 2019. [En línea]. Available: https://www.amazon.es/SEGA-Mega-Drive-Mini-Controllers/dp/B07S8XVWB8. [Último acceso: 29 de Abril 2020].

[19] Canalrcn, «Canalrcn,» 22 de Diciembre 2019. [En línea]. Available: https://www.canalrcn.com/todogamers/resenas-y-analisis/articulo-nota/virtual-boy-el-mayor-fracaso-que-tuvo-nintendo-en-toda-su-historia. [Último acceso: 29 de Abril 2020].

[20] I. T. d. Aragón, «Análisis: Motores gráficos y su aplicación en la industria,» Tecsmedia, Aragón, 2014.

[21] U. Ruelas, «Codingornot,» 13 de Julio 2017. [En línea]. Available: https://codingornot.com/que-es-un-motor-de-videojuegos-game-engine. [Último acceso: 29 de Abril 2020].

[22] A. Tecnoempleo, «Tecnoempleo,» 14 de Noviembre 2018. [En línea]. Available: https://blog.tecnoempleo.com/candidatos/2018/13242/trabajar-en-el-mundo-de-los-videojuegos-unity-or-unreal/. [Último acceso: 29 de Abril 2020].

[23] Cryengine, «Cryengine,» 2020. [En línea]. Available: https://www.cryengine.com/support/view/general#. [Último acceso: 29 de Abril 2020].

[24] A. Soloaga, «Akademus,» 19 de Julio 2019. [En línea]. Available: https://www.akademus.es/blog/emprendedores/unreal-engine-que-es-y-para-que-sirve/. [Último acceso: 29 de Abril 2020].

[25] R. King, «Bitdegree,» 23 de Enero 2020. [En línea]. Available: https://es.bitdegree.org/tutoriales/crear-videojuegos/. [Último acceso: 29 de Abril 2020].

[26] Iboxacademy, «Iboxacademy,» 2019. [En línea]. Available: https://Iboxacademy.es/blog/diferencias-entre-unity-y-unreal/. [Último acceso: 29 de Abril 2020].

[27] J. Peña, «arpentechnologies,» 12 de Febrero 2018. [En línea]. Available: https://arpentechnologies.com/es/blog/software/principales-motores-graficos-para-indies/. [Último acceso: 29 de Abril 2020].

[28] Lobo7922, «Cuevadelobo,» 19 de Enero 2018. [En línea]. Available: https://www.cuevadelobo.com/ventajas-desventajas-unreal-engine-4/. [Último acceso: 29 de Abril 2020].

[29] Barritico, «Forum.unity,» 9 de Junio 2017. [En línea]. Available: https://forum.unity.com/threads/high-cpu-usage.581494/. [Último acceso: 29 de Abril 2020].

[30] PugetSystems, «PugetSystems,» 2020. [En línea]. Available: https://www.pugetsystems.com/recommended/Recommended-Systems-for-Unreal-Engine-200/Hardware-Recommendations. [Último acceso: 29 de Abril 2020].

[31] 3djuegos, «3djuegos,» 2020. [En línea]. Available: https://www.3djuegos.com/juegos/requisitos/15972/cryengine/. [Último acceso: 29 de Abril 2020].

[32] SMICHRISSOFT, «Medium,» 03 de Junio 2018. [En línea]. Available: https://medium.com/@smichrissoft/unity-vs-unreal-engine-for-2d-and-3d-games-development-72e4e6893fbd. [Último acceso: 29 de Abril 2020].

[33] F. 5. T. Hub, «factoria5hub,» 14 de Agosto 2019. [En línea]. Available: https://www.factoria5hub.com/single-post/10TerminosUE4. [Último acceso: 29 de Abril 2020].

[34] A. Muciño, «Hardwareviews,» 23 de Marzo 2016. [En línea]. Available: https://hardwareviews.com/cryengine-v-el-nuevo-motor-grafico-de-crytek/. [Último acceso: 29 de Abril 2020].

[35] «Store.unity,» 29 de Abril 2020. [En línea]. Available: https://store.unity.com/es/. [Último acceso: 29 de Abril 2020].

[36] Poor-man, «Geektopia,» 16 de Marzo 2016. [En línea]. Available: https://www.geektopia.es/es/technology/2016/03/16/noticias/el-motor-de-videojuegos-cryengine-se-pasa-al-modelo-paga-lo-que-tu-quieras.html. [Último acceso: 29 de Abril 2020].

[37] Wikipedia, «Wikipedia,» 26 de Marzo 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Historia_de_los_videojuegos. [Último acceso: 29 de Abril 2020].

[38] «Fib,» [En línea]. Available: https://www.fib.upc.edu/retro-informatica/historia/videojocs.html. [Último acceso: 29 de Abril 2020].

[39] «Qwant,» [En línea]. Available: https://www.qwant.com/game/tennis-for-two?l=nl. [Último acceso: 29 de Abril 2020].

[40] A. o. I. A. &. Aciences, «Interactive.org,» 2020. [En línea]. Available: https://www.interactive.org/news/112018_spacewar_pioneer.asp. [Último acceso: 29 de Abril 2020].

[41] L. E. P. García, «Monografias.com,» 2012. [En línea]. Available: https://www.monografias.com/trabajos90/historia-videojuegos/historia-videojuegos.shtml. [Último acceso: 29 de Abril 2020].

[42] S. Torres, «Puntodebreak,» 13 de Septiembre 2014. [En línea]. Available: http://www.puntodebreak.com/2014/09/13/pong-el-precursor-de-los-juegos-de-tenis. [Último acceso: 29 de Abril 2020].

[43] A. García, «Lavanguardia,» 21 de Enero 2020. [En línea]. Available: https://www.lavanguardia.com/tecnologia/20180725/451089770519/space-invaders-40-aniversario-marcianitos-arcade.html. [Último acceso: 29 de Abril 2020].

[44] S. Hemmer, «HubPages,» 31 de Enero 2013. [En línea]. Available: https://hubpages.com/games-hobbies/Evolution-of-Breakout. [Último acceso: 29 de Abril 2020].

[45] A. CD, «Vidaextra,» 29 de Junio 2013. [En línea]. Available: https://www.vidaextra.com/cultura/el-moma-expone-la-magnavox-odyssey-y-anade-seis-juegos-mas-a-su-coleccion. [Último acceso: 29 de Abril 2020].

[46] J. Rosenberg, «Thoughtco,» 24 de Marzo 2020. [En línea]. Available: https://www.thoughtco.com/pac-man-game-1779412. [Último acceso: 29 de Abril 2020].

[47] Alteredgamer, «Alteredgamer,» [En línea]. Available: https://www.alteredgamer.com/football-manager/68198-the-history-of-football-management-games-part-2/. [Último acceso: 29 de Abril 2020].

[48] A. Wilson, «Funstockretro,» 27 de Octubre 2015. [En línea]. Available: https://www.funstockretro.co.uk/news/miner-49er-49-facts-about-manic-miner/. [Último acceso: 29 de Abril 2020].

[49] Nintendo, «Nintendo,» 2020. [En línea]. Available: https://www.nintendo.es/Empresa/La-historia-de-Nintendo/La-historia-de-Nintendo-625945.html. [Último acceso: 29 de Abril 2020].

[50] Mobygames, «Mobygames,» 2017. [En línea]. Available: https://www.mobygames.com/game/knight-lore. [Último acceso: 29 de Abril 2020].

[51] M. Weisberger, «Livescience,» 13 de Octubre 2016. [En línea]. Available: https://www.livescience.com/56481-strange-history-of-tetris.html. [Último acceso: 29 de Abril 2020].

[52] Nintendo, «Nintendo,» 2020. [En línea]. Available: https://mario.nintendo.com/history/. [Último acceso: 29 de Abril 2020].

[53] Computinghistory, «Computinghistory,» 2016. [En línea]. Available: http://www.computinghistory.org.uk/det/4034/Sega-Master-System/. [Último acceso: 29 de Abril 2020].

[54] chrisscullion, «Tiredoldhack,» 26 de Mayo 2018. [En línea]. Available: https://tiredoldhack.com/2018/05/26/the-complete-history-of-street-fighter/. [Último acceso: 29 de Abril 2020].

[55] 16bitdad, «16bitdad,» 14 de Abril 2018. [En línea]. Available: https://www.16bitdad.com/blog/the-history-of-final-fantasy/. [Último acceso: 14 04 2020].

[56] Cuantos.org, «Cuantos,» 2015. [En línea]. Available: http://www.cuantos.org/c368/Cuantas-consolas-Master-System-se-han-vendido. [Último acceso: 29 de Abril 2020].

[57] R. MÁRQUEZ, «xataka,» 28 de Diciembre 2013. [En línea]. Available: https://www.xataka.com/videojuegos/nes-ya-hemos-puesto-a-prueba-y-flipado-con-la-consola-definitiva-de-nintendo. [Último acceso: 29 de Abril 2020].

[58] Pngio, «Pngio,» [En línea]. Available: https://pngio.com/images/png-a693998.html. [Último acceso: 29 de Abril 2020].

[59] Espaciopsicofamiliar, «Espaciopsicofamiliar,» [En línea]. Available: https://espaciopsicofamiliar.es/pac-man-estimula-nuestro-cerebro-desde-1980/. [Último acceso: 29 de Abril 2020].

[60] Crashonline, «Crashonline,» [En línea]. Available: http://www.crashonline.org.uk/12/knghtlr.htm. [Último acceso: 29 de Abril 2020].

[61] M. Gerardi, «News.avclub,» 9 de Agosto 2015. [En línea]. Available: https://news.avclub.com/shigeru-miyamoto-explains-the-making-of-super-mario-bro-1798284041. [Último acceso: 29 de Abril 2020].

[62] A. Gomes, «Centraldeheroes,» 30 de Agosto 2018. [En línea]. Available: http://centraldeheroes.com/street-fighter-1-aniversario-31/. [Último acceso: 29 de Abril 2020].

[63] E. Revert, «Lavanguardia,» 06 de Junio 2019. [En línea]. Available: https://www.lavanguardia.com/historiayvida/mas-historias/20190530/47312506910/tetris-asi-empezo-todo.html. [Último acceso: 29 de Abril 2020].

[64] «Computinghistory,» [En línea]. Available: http://www.computinghistory.org.uk/det/24546/Super-Nintendo-Entertainment-System/. [Último acceso: 29 de Abril 2020].

[65] Gallego, «Vidaextra,» 13 de Junio 2008. [En línea]. Available: https://www.vidaextra.com/pc/especial-repasamos-la-saga-alone-in-the-dark. [Último acceso: 29 de Abril 2020].

[66] M. Minotti, «Venturebeat,» 28 de Mayo 2014. [En línea]. Available: https://venturebeat.com/2014/05/28/the-complete-history-of-mario-kart/. [Último acceso: 29 de Abril 2020].

[67] Mobygames, «Mobygames,» [En línea]. Available: https://www.mobygames.com/game/simon-the-sorcerer. [Último acceso: 29 de Abril 2020].

[68] W3M4R1S0L, «Slideshare,» 12 de Mayo 2013. [En línea]. Available: https://es.slideshare.net/W3MAR1S0L/sony-21056138. [Último acceso: 29 de Abril 2020].

[69] Jarkendia, «Vidaextra,» 15 de Octubre 2017. [En línea]. Available: https://www.vidaextra.com/estrategia/20-anos-de-age-of-empires-asi-es-la-historia-de-uno-de-los-rts-mas-laureados-del-pc. [Último acceso: 29 de Abril 2020].

[70] Tomaydame, «Tomaydame,» [En línea]. Available: https://www.tomaydame.es/producto/99625/NINTENDO-NINTENDO-64-CONTROL-DECK. [Último acceso: 29 de Abril 2020].

[71] L. Raffa, «Vix,» [En línea]. Available: https://www.vix.com/es/videojuegos/170961/los-20-juegos-mas-vendidos-en-la-historia-del-playstation. [Último acceso: 29 de Abril 2020].

[72] Soujiro, «Los40,» 25 de Noviembre 2019. [En línea]. Available: https://los40.com/los40/2019/11/25/tecnologia/1574677264_082805.html. [Último acceso: 29 de Abril 2020].

[73] K. Kalata, «Hardcoregaming101,» 12 de Mayo 2008. [En línea]. Available: http://www.hardcoregaming101.net/secret-of-monkey-island-the/. [Último acceso: 29 de Abril 2020].

[74] jaimixx, «Insertcoinclasicos,» 14 de Agosto 2006. [En línea]. Available: https://www.insertcoinclasicos.com/2006/08/14/sid-meier-s-civilization-1991-/. [Último acceso: 29 de Abril 2020].

[75] H. Todd, «Gamasutra,» 17 de Febrero 2015. [En línea]. Available: https://www.gamasutra.com/blogs/HamishTodd/20150217/236516/Level_design_Dooms_quothorseshoequot.php. [Último acceso: 29 de Abril 2020].

[76] P. R. Donoso, «startvideojuegos,» 09 de Septiembre 2015. [En línea]. Available: http://www.startvideojuegos.com/dos-consolas-dos-caminos/. [Último acceso: 29 de Abril 2020].

[77] Computinghistory, «Computinghistory,» [En línea]. Available: http://www.computinghistory.org.uk/det/4597/Sony-Playstation-2/. [Último acceso: 29 de Abril 2020].

[78] C. McFadden, «Interestingengineering,» 24 de Octubre 2019. [En línea]. Available: https://interestingengineering.com/history-of-microsoft-and-its-major-milestones. [Último acceso: 29 de Abril 2020].

[79] Historycooperative, «Historycooperative,» [En línea]. Available: https://historycooperative.org/the-history-of-the-iphone/. [Último acceso: 29 de Abril 2020].

[80] B. O'Boyle, «Pocket-lint,» 11 de Abril 2020. [En línea]. Available: https://www.pocket-lint.com/tablets/news/apple/146888-history-of-the-apple-ipad. [Último acceso: 29 de Abril 2020].

[81] A. CD, «Vidaextra,» 04 de Octubre 2018. [En línea]. Available: https://www.vidaextra.com/nintendo-switch/nintendo-planea-lanzar-nueva-version-switch-2019-wsj. [Último acceso: 29 de Abril 2020].

[82] elotrolado, «elotrolado,» [En línea]. Available: https://www.elotrolado.net/wiki/PlayStation_4. [Último acceso: 29 de Abril 2020].

[83] M. Bravo, «FayerWayer,» 15 de Abril 2020. [En línea]. Available: https://www.fayerwayer.com/2020/04/xbox-one-x-baja-su-precio/. [Último acceso: 29 de Abril 2020].

[84] Nintendo, «Nintendo,» 07 de Diciembre 2018. [En línea]. Available: https://www.nintendo.com/es_LA/games/detail/super-smash-bros-ultimate-switch/. [Último acceso: 29 de Abril 2020].

[85] fortniteros, «fortniteros,» 01 de Octubre 2017. [En línea]. Available: https://www.fortniteros.es/vlog/gameplays/fortnite-gameplay-highlights/. [Último acceso: 29 de Abril 2020].

[86] J. Horwitz, «Venturebeat,» 13 de Julio 2018. [En línea]. Available: https://venturebeat.com/2018/07/13/sony-japan-studio-interview-how-astro-bots-vr-changes-3d-platforming/. [Último acceso: 29 de Abril 2020].

[87] mmos, «mmos,» 2016. [En línea]. Available: https://mmos.com/review/clash-royale. [Último acceso: 29 de Abril 2020].

[88] R. Blog, «Medium,» 20 de Abril 2018. [En línea]. Available: https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a. [Último acceso: 29 de Abril 2020].

[89] Autodesk, «Autodesk,» 2020. [En línea]. Available: https://www.autodesk.es/products/3ds-max/overview. [Último acceso: 29 de Abril 2020].

[90] Unity, «Unity3d,» 2020. [En línea]. Available: https://docs.unity3d.com/Manual/index.html. [Último acceso: 29 de Abril 2020].

[91] I. t. d. Durango, «Manual de 3ds MAX,» Durango, 2018.

[92] M. P. Pons, «Mondoiberica,» 27 de Junio 2018. [En línea]. Available: https://news.mondoiberica.com.es/campos-futbol-sant-lluis-menorca-dual-mondo/. [Último acceso: 29 de Abril 2020].

[93] Surfingtheplanet, «Surfingtheplanet,» 2019. [En línea]. Available: https://www.surfingtheplanet.com/que-ver-provenza-lugares-imprescindibles/. [Último acceso: 29 de Abril 2020].

[94] República/EFE, «Republica,» 08 de Enero 2018. [En línea]. Available: https://www.republica.com/2018/01/08/descubren-una-de-las-galaxias-mas-antiguas-del-universo/. [Último acceso: 29 de Abril 2020].

[95] L. Valeiro, «Elmundo,» 09 de Julio 2018. [En línea]. Available: https://www.elmundo.es/vida-sana/bienestar/2018/07/09/5b3f4e10268e3ecb6f8b4575.html. [Último acceso: 29 de Abril 2020].

[96] WikiWand, «WikiWand,» [En línea]. Available: https://www.wikiwand.com/es/Anexo:Cuerpos_de_agua_de_la_Región_Metropolitana_de_Santiago. [Último acceso: 29 de Abril 2020].

[97] turismoruralnavarra, «turismoruralnavarra,» 2020. [En línea]. Available: https://www.turismoruralnavarra.com/tour-item/comarca-de-pamplona/. [Último acceso: 29 de Abril 2020].

[98] tilytravels, «tilytravels,» 13 de Noviembre 2019. [En línea]. Available: https://www.tilytravels.com/blog/a-guide-to-choosing-the-best-vacation-rental-for-your-trip-to-lake-placid. [Último acceso: 29 de Abril 2020].

[99] Jessy, «fondos.wallpaperstock,» 21 de Julio 2015. [En línea]. Available: https://fondos.wallpaperstock.net/niza-bosque-acantilado-campos-de-río-wallpapers_w49198.html. [Último acceso: 29 de Abril 2020].

[100] pinterest, «pinterest,» 2020. [En línea]. Available: https://www.pinterest.es/pin/24840235429497797/. [Último acceso: 29 de Abril 2020].

[101] k-life, «Freepik,» 2020. [En línea]. Available: https://www.freepik.es/fotos-vectores-gratis/textura-tierra. [Último acceso: 29 de Abril 2020].

[102] party-city, «party-city,» 2020. [En línea]. Available: https://www.party-city.es/Producto-Detalle.aspx?ProductID=HOCUWAND. [Último acceso: 29 de Abril 2020].

[103] disfracessimon, «disfracessimon,» 2020. [En línea]. Available: https://www.disfracessimon.com/complementos-disfraz/4113-varita-magica-plateada.html. [Último acceso: 29 de Abril 2020].

[104] fnac, «fnac.es,» 2020. [En línea]. Available: https://www.fnac.es/a6191166/Varita-magica-Harry-Potter-con-hechizos. [Último acceso: 29 de Abril 2020].

[105] amazon, «amazon,» 2020. [En línea]. Available: https://www.amazon.co.uk/Cosplay-Keychain-Necklace-Dumbledore-Voldemort/dp/B07PP2JMZX. [Último acceso: 29 de Abril 2020].

[106] ldoceonline, «ldoceonline,» 2020. [En línea]. Available: https://www.ldoceonline.com/es-LA/dictionary/pine. [Último acceso: 29 de Abril 2020].

[107] encinasorigenehistoria, «encinasorigenehistoria.blogspot.com,» 27 de Septiembre 2017. [En línea]. Available: http://encinasorigenehistoria.blogspot.com/2015/02/quercus-este-genero-tiene-mas-de-150.html. [Último acceso: 29 de Abril 2020].

[108] A. E. Ruiz, «historiadelavida,» [En línea]. Available: https://historiadelavida.editorialaces.com/tipos-de-rocas/. [Último acceso: 29 de Abril 2020].

[109] pinterest, «pinterest,» 2020. [En línea]. Available: https://www.pinterest.es/pin/559150109971337190/. [Último acceso: 29 de Abril 2020].

[110] Agremaqcr, «agremaqcr,» 08 de Enero 2019. [En línea]. Available: https://agremaqcr.com/la-roca-nuestra-materia-prima/. [Último acceso: 29 de Abril 2020].

[111] R. Quelart, «lavanguardia,» 24 de Octubre 2019. [En línea]. Available: https://www.lavanguardia.com/vida/20191024/471168657010/setas-toxicas-mortales-hospital-clinic-intoxicaciones.html. [Último acceso: 29 de Abril 2020].

[112] isaro, «isaro,» 28 10 2019. [En línea]. Available: https://www.isaro.com/noticias-d.php?Id=537. [Último acceso: 29 de Abril 2020].

[113] G. Portillo, «jardineriaon,» 2020. [En línea]. Available: https://www.jardineriaon.com/setas-venenosas.html. [Último acceso: 29 de Abril 2020].

[114] A. A. E. d. Videojuegos, «LA INDUSTRIA DEL VIDEOJUEGO EN ESPAÑA Anuario 2018,» 2018.

[115] kriplus, «kriplus.es,» 29 de Abril 2020. [En línea]. Available: https://www.kriplus.es/portatiles/portatil-asus-gl552v-intel-core-i7-6700hq-156-8-gb-de-ram-disco-duro-1-tb-geforce-gtx-960m-2-gb-windows-10-4712900176483.html. [Último acceso: 29 de Abril 2020].

[116] Amazon, «Amazon,» 2020. [En línea]. Available: https://www.amazon.es/htc-vive/s?k=htc+vive. [Último acceso: 29 de Abril 2020].

[117] bit4learn, «bit4learn,» 2020. [En línea]. Available: https://bit4learn.com/es/lms/blackboard/. [Último acceso: 29 de Abril 2020].

[118] microsoft, «microsoft,» 29 de Abril 2020. [En línea]. Available: https://www.microsoft.com/es-es/store/b/windows. [Último acceso: 29 de Abril 2020].

[119] office, «office,» 2020. [En línea]. Available: https://products.office.com/es-ES/home/. [Último acceso: 29 de Abril 2020].

[120] GAndroid, «galaxyandroid,» 13 de Enero 2014. [En línea]. Available: https://www.galaxyandroid.es/cual-es-el-consumo-de-un-ordenador-pc-o-portatil/. [Último acceso: 29 de Abril 2020].

[121] tarifasgasluz, «tarifasgasluz,» 20 de Marzo 2020. [En línea]. Available: https://tarifasgasluz.com/comercializadoras/iberdrola/precio-kwh. [Último acceso: 29 de Abril 2020].

ANEXO A. GLOSARIO

Acrónimos

2D	*2 dimensiones*
3D	*3 dimensiones*
AEVI	*Asociación Española de Videojuegos*
CDROM	*Compact Disc Read-Only Memory*
CPU	*Central Processing Unit*
ENIAC	*Electronic Numerical Integrator And Computer*
HTML	*HyperText Markup Language*
IBM	*International Business Machines*
IOS	*iPhone Operating System*
MAC OS	*Macintosh Operating System*
MB	*Mega Byte*
MEMEX	*Memory - Index*
NASA	*National Aeronautics and Space Administration*
OS	*Operating System*
OSVR	*Open Source Virtual Reality*
PEGI	*Pan European Game Information*
PS4	*PlayStation 4*
PSVR	*PlayStation Virtual Reality*
RAM	*Random Access Memory*
TFG	*Trabajo Fin de Grado*
TX	*Transistorized Experimental Computer*
VR	*Virtual Reality*

Términos

Android	Sistema operativo de dispositivos móviles.
Atari	Empresa impulsora principalmente de videojuegos arcade y ordenadores personales.
Autodesk 3DS MAX	Programa de diseño, modelado y animación de objetos 2D y 3D.
Blizzard Entertainment	Empresa impulsora de videojuegos mayormente de estrategia en tiempo real.
CSharp	Lenguaje de programación derivado de C++.
C++	Lenguaje de programación híbrido en programación orientada a objetos y manipulación de objetos.
Capcom	Empresa impulsora fundamentalmente de videoconsolas y videojuegos arcade.
Collider	Área que encierra un objeto para permitirle colisiones.
Hardware	Conjunto de componentes físicos que conforman un sistema o computador.
JavaScript	Lenguaje de programación principalmente utilizado para el desarrollo de páginas webs y animaciones 2D y 3D.
LucasFilm Games	Empresa impulsora de aventuras cinematográficas en los videojuegos.
Microsoft	Empresa dedicada al desarrollo de sistemas software y hardware electrónico.
Nintendo	Empresa impulsora de videoconsolas y videojuegos innovadores para la época.
Plugin	Software creado para ser utilizado en un programa o conjunto de programas con el propósito de mejorar el conjunto de programas o dar facilidades al usuario.
Rol en videojuegos	Género de videojuegos donde el jugador puede controlar un personaje, cambiar sus características e interactuar con otros personajes no jugables dentro de un mundo.
Sega	Empresa dedicada primordialmente al desarrollo de videojuegos.
Software	Conjunto de componentes lógicos que posibilitan la ejecución de una tarea.
Sony	Empresa impulsora de videoconsolas, videojuegos y teléfonos móviles innovadores para la época.
Square Enix	Empresa impulsora principalmente de videojuegos rol.
Supercell	Empresa dedicada al desarrollo de videojuegos para móviles.
Taito Corporation	Empresa dedicada fundamentalmente al desarrollo de videojuegos.

ANEXO B. MANUAL DE USO

Manual de usuario

El propósito de esta sección es explicar los pasos necesarios para jugar al videojuego desarrollado. Primero se deberá de instalar los visores correctamente y, posteriormente se deberá de abrir el ejecutable con los visores puestos. Finalmente, se explicará cómo jugar al videojuego.

Preparativos para el arranque del videojuego

El paso inicial para jugar al videojuego será conectar los visores de realidad virtual.

HTC Vive

Las HTC Vive cuentan con dos estaciones base, dos mandos y los visores. Lo primero que debe realizarse es colocar las dos estaciones base en esquinas opuestas dentro de la habitación donde se desea jugar. La altura mínima de cada estación base es de 2 metros.

La conexión de las estaciones base puede ser por cable físico de sincronización o sin cable. En caso de utilizar cable, una base deberá encontrarse en el canal “A” y otra en el “b”, en caso de no utilizar el cable, una base deberá encontrarse en el canal “b” y la otra en el “c”.

Posteriormente, se deben enchufar los visores al ordenador. Para que los visores sean detectados, se deberá de acceder a la plataforma “Steam” para descargar la aplicación “Steam VR”. Finalmente, se ejecutará “Steam VR” para instalar los visores siguiendo el tutorial que la propia aplicación te muestra.

Arranque del videojuego

El usuario deberá descomprimir el archivo “Videojuego_Optimizado” o “Videojuego_No_Optimizado” si posee un equipo con elevados recursos o no respectivamente. Entonces el usuario deberá abrir el directorio descomprimido.

Para abrir el videojuego deberá seleccionar el archivo “VideojuegoVirtual.exe”.

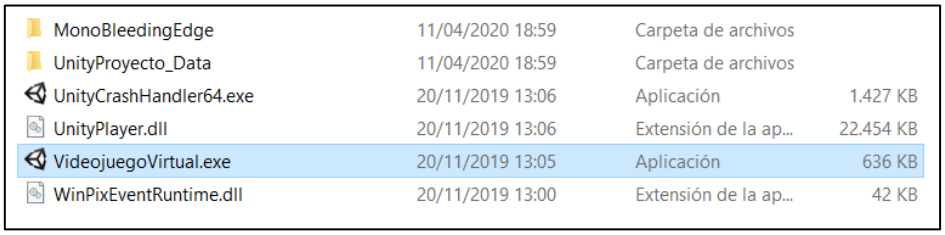


Fig.161 -- Ejecución del videojuego

El juego se iniciará automáticamente. Por tanto, podrá coger los mandos y colocarse los visores en la cabeza para dar comienzo con su experiencia con el juego.

Cómo jugar

Los controles que posee el usuario en el videojuego son desplazamiento del jugador tanto agacharse como saltar (dentro de las limitaciones de la realidad virtual) y el movimiento de los brazos de acuerdo con el de los mandos.

La primera escena que se muestra al usuario es la del menú principal. Con la mirada al frente observará principalmente 4 pantallas. Las 3 pantallas inferiores se utilizarán para modificar los parámetros de la prueba, en concreto, la velocidad de los obstáculos, la duración de la prueba y el intervalo de salida. El usuario puede pulsar el botón verde para incrementar el valor de estos parámetros y el botón rojo para decrementarlos. Para el caso de la duración de la prueba, los valores de modificación serán de 20 unidades mientras que en el resto de los parámetros será de 1 unidad. La pantalla superior mostrará un breve resumen de cómo jugar que sirve como recordatorio.

En cada lateral del usuario se encuentran dos pantallas de iniciar partida, la pantalla de la izquierda muestra el botón naranja que inicia la partida con los parámetros seleccionados en el escenario simple y la pantalla del lateral derecho conduce al escenario complejo.

Tras finalizar la partida, en el mismo directorio donde se encuentra el ejecutable del videojuego se encontrará el fichero “resultados.txt” con los resultados de la prueba.

SUMMARY

This chapter corresponds to the english summary that must be included in the 2011 plan computer engineering degree. The summary is structured in introduction and objectives, design of the solution, experiments and conclusions.

INTRODUCTION AND GOALS

This work searches to create a virtual reality videogame prototype to train motor coordination in patients with neurological disorders.

Introduction

Neurological disorders are diseases that affect the nervous system and the brain. The symptoms of a neurological disorder are very varied depending on the type of person and the disease. Generally, the disorders alter the mobility and perception of the senses causing loss of balance, paralysis, inability to speak... The duration of these periods is very varied depending on the phase in which it is.

Diagnosing and treating the patient as early as possible are the most important factors for a patient's recovery. This work will focus on creating a rehabilitation exercise to facilitate that patient's recovery process.

Goals and motivation

The motivation for this project comes mainly from people who are suffering from neurological disorders. In addition, virtual reality has improved since its origins and currently manages to obtain realistic simulations to reach different areas: education, psychotherapy, driving ... The main objectives of this work are:

- Learn how to use the Unity graphics engine to develop the functionality and animation of the software system.
- A Learn how to use the Autodesk 3DS MAX program to develop the modeling of the three-dimensional objects used.

The student's videogame must also comply other objectives:

- Computers are improving their performance over the years and today, every home has at least one computer so, the videogame must allow gameplay for the pc.
- The fluency of a videogame affects the degree of conformity of the player so, the videogame must be fluid.
- The videogame must not present failures that block its progress.
- The art used for the scenes must be pleasant and the player must understand each element of the scene.
- The videogame must have a main menu interface before testing.
- The interface must be intuitive for the end user even if the user does not have experience in videogames.

DESIGN OF THE SOLUTION

Software, techniques and methods proposed

With the analysis of the art state and the gathering of requirements in section "2. PLANTEAMIENTO DEL PROBLEMA", Unity was chosen as a graphics engine for the technical solution.

Design and modeling of 3D objects: it consists of the modeling of all 3D objects that are shown in the scenes of the videogame. The student has decided to design and model all three-dimensional objects.

3DS MAX enables the generation of two-dimensional and three-dimensional graphics, objects, and animations. This program offers a free 3-year license to students. In addition, it is very easy to install free plugins that incorporate pre-defined animations and models. The student will use a plugin only to learn how to use them, however, they will not use any plugin in their final models to get as much instruction as possible about this program.

The objects modeled and textured with this program will be exported and later imported into the Unity program.

Programming and animation of a videogame: it consists about the creation of the complete scenes of the videogame. This program will allow the importation the objects into the scene, insert background music, texturization of different elements, position of all three-dimensional objects, animation of objects and programming the functionality.

Different techniques and methods have been used:

- Scripting: consists of the creation of "scripts" files. These files are used to implement the operation of the objects with the selected programming language. In this case, in C Sharp language.
- Drag: consists of dragging an item to a destination. It has been primarily used in Unity when moving objects between directories or dragging objects into the scene.
- Positioning: consists of moving the three-dimensional objects around the scene modifying their coordinates.
- Steam VR: Steam VR: consists of a free downloadable package from the "asset store" that can insert directly a player with the basic operation of virtual reality: check the controls and the viewfinder, playback in the viewers, detection of the player and his displacement on the real environment ...

Final design selected

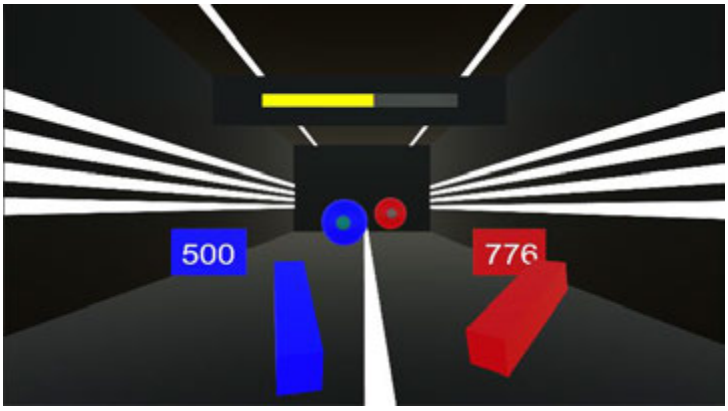
A videogame design was chosen with a study of design alternatives. Finally, the videogame will show a sequence of obstacles that approach the player, the player must touch the obstacle as close to the center as possible. A representative color will be used for each hand to distinguish each obstacle.

Simple scenario development

This scenario is called “simple” because it is made of polyhedrons except for sphere shaped obstacles. The colors used are flat. This scenario is very useful to obtain

the dimensions of the obstacles, the starting position of the player and the distance between the player's starting position and the position of the obstacles. Also, it serves as an introduction to learning the programs.

The player will first use this scenario to train and understand the operation of the videogame.

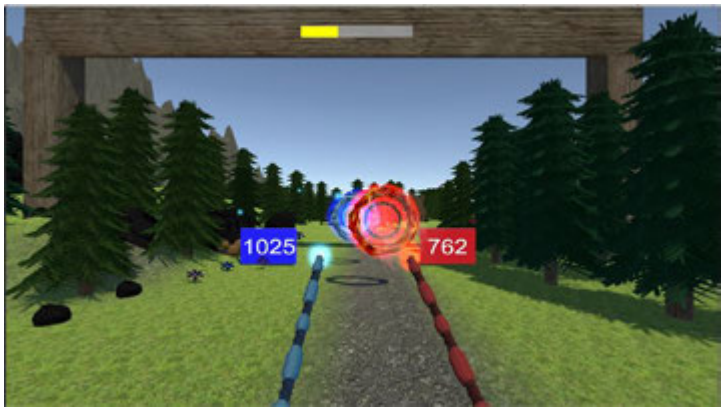


The hand corresponding to each obstacle is perfectly distinguishable. The yellow bar represents the remaining time of the test and the numbers indicate the score reached at the moment with each hand.

Complex scenario development

This scenario is called "complex" because the objects modeled in the environment have more elaborate shapes. Keep in mind that the player is a patient who has suffered a neurological disorder therefore, the environment should represent a calm and friendly environment for the player.

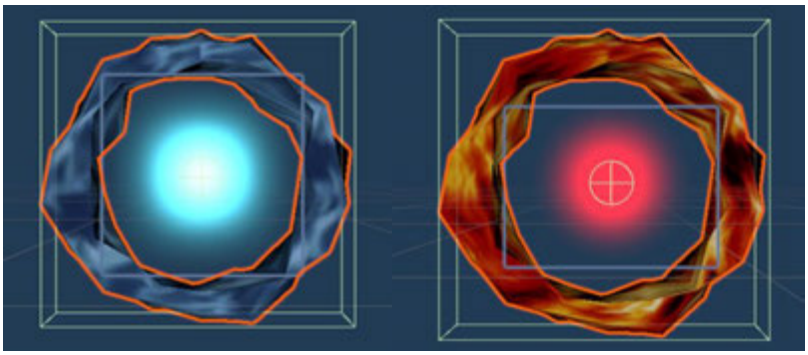
The finally selected environment is a forest because it calms the player and encourages his concentration. For years, magic forests have been used in videogames and movies.



This design does not confuse the player because it maintains the same essence as the simple scenario.

Physics in functional objects

The property "RigidBody" will be included in the sphere-shaped obstacles to keep them under the laws of physics. But the gravitational forces will be disabled.

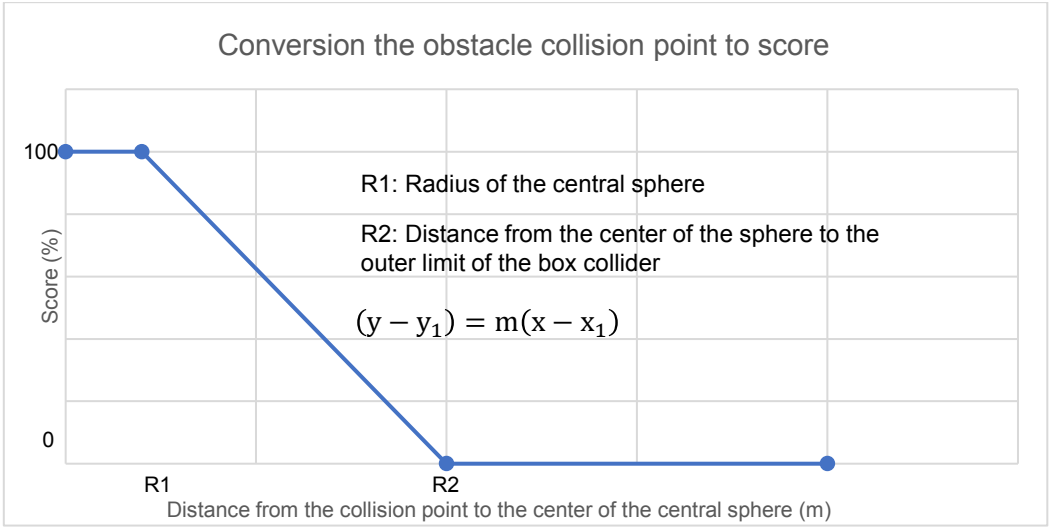


Two collision zones (collider) will also be added, the first box-shaped detects the collision anywhere on the body and the second sphere-shaped represents the center of the obstacle.

For magic wands two colliders will be used to wrap the entire body.

Score calculation per hit

Each obstacle has a central spherical region that represents the center of itself. If the player hits the central region then he will get 100% accuracy, if the hit is outside the central region then the distance to the center of the region is calculated to get the score.



Data storage and interest parameters

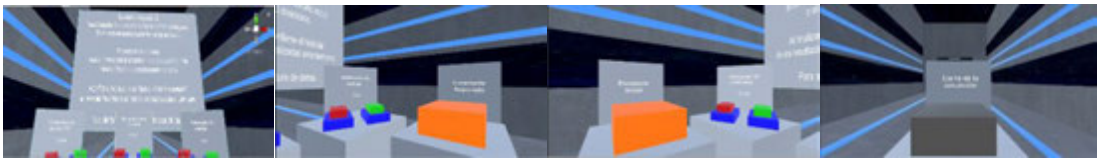
The system maintains a list of all obstacles created during the game. When the player hits an obstacle, the obstacle reference is removed from the list. At the end of the game, this list contains the obstacles not reached by the user.

The system keeps stored all the results obtained by the patient in the game: the total number of obstacles of each color, the total number of obstacles hit of each color, the total number of non-hit obstacles of each color, the total number of obstacles incorrectly hit of each color and the player's total precision with the obstacles of each color. The results obtained will be automatically written to the file "resultados.txt".

Main menu creation

The main menu consists of the initial scenario where the game begins. The purpose of the main menu is to give the user the possibility to modify the system parameters: speed of the obstacles, time interval between each obstacle and duration of the game. The player can also choose the scenario (simple or complex) where to use the parameter values.

The developed environment is the interior of a spaceship.

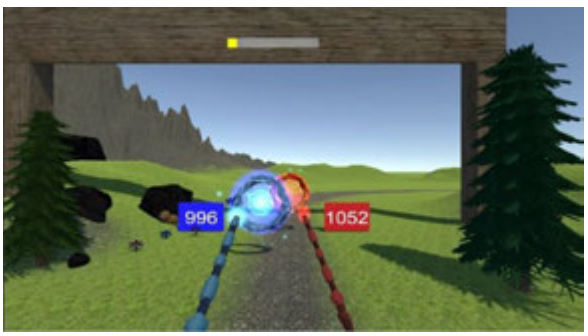


The user has different buttons to modify the parameters of the test. There are also text screens to help the user to understand how each button works.

Optimization

The main disadvantage of virtual reality is the high consumption of system resources. In addition, if you have a computer at the limit of the minimum requirements of virtual reality (is the case of the student), the load of the numerous components will mean an increase in latency and a decrease in the fluidity of the videogame.

In the forest, the number of trees negatively influences the fluidity of the videogame, which is a discontent in the user experience. It has been decided to create an optimization of the forest scenario. Then the number of decorative elements such as trees will be reduced.



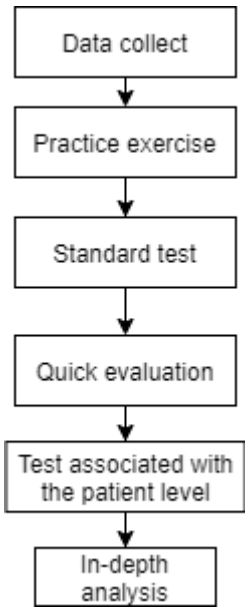
Finally, the two versions of the videogame will be published: the optimized and the non-optimized. Thus, both users with good resources and those with poorer resources will be able to play fluently enough.

EXPERIMENTS

Performed protocol

The optimized version has been used to carry out the tests. In this way, the fluidity of the videogame is guaranteed for all tests.

The procedure followed with each patient is:



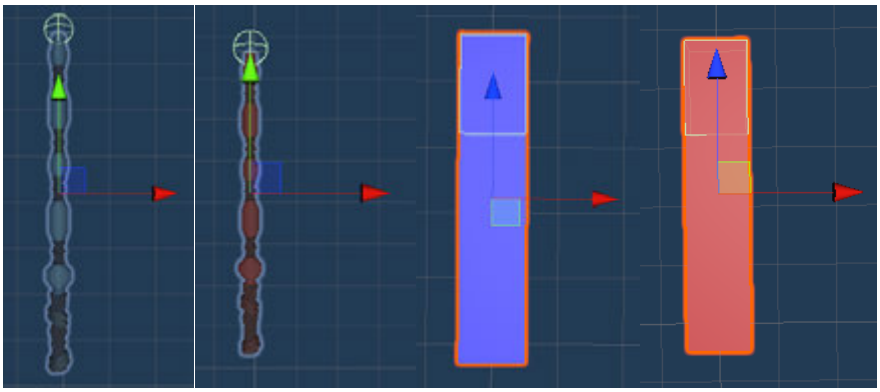
1. Patient utility data is recorded.
2. The patient plays one game with standard conditions in the simple scenario and another in the complex scenario. These conditions are test duration 100s, obstacle speed 3u / s and time interval between each obstacle 3s.
   The results of these games will not be taken into account. The purpose is to train the patient.
3. The patient plays a standard game in the simple scenario and another in the complex scenario.
4. Therapists analyze the results obtained by the patient and choose new parameter values.
5. The patient plays the new test in the simple and complex scenario with the parameters selected by the therapists.
6. Therapists analyze in the results obtained by the patients.

Non-protocol test

These tests found a 62-year-old man with no experience in virtual reality and difficulties in moving his right arm.

The student brought the virtual reality equipment personally to the user's home. Proper placement of all devices is important to increase the feeling of virtual immersion. In this case, the tests were performed in the user's dining room and the best possible calibration of the controls and displays was not obtained, therefore, the results obtained have been influenced. On the other hand, the user has problems in the right part of the body, this implies that he only played the videogame with his left hand.

On some occasions, the user avoided the obstacles with an unexpected behavior on the part of the student. Intuitively, if the user observed an obstacle to be avoided, he moved the saber/wand downwards to not hit the tip with the obstacle (but the obstacle must not hit any part of the saber/wand). Then, the system detected that the user was hitting obstacles when the user intended to move away. For this reason, it has been decided to modify the collision region of the saber and wand. In summary, approximately 30% of the upper part of the saber is considered as an adjoining region and for the wand only the upper sphere, the rest of the object will not affect upon contact with the obstacle. With this, you can avoid the obstacles by moving the tip of the saber/wand and not the entire object.



Errors were also found in the destroyed red obstacle counter. When a red obstacle was destroyed, the blue destroyed obstacles counter increased. Therefore, the results show 0 hit red obstacles, but the patient was seen to make mistakes hitting red obstacles incorrectly. On the other hand, when a test is started, the number of obstacles of each color is not known because their generation is pseudo-random, so it is at the end of the test that the conditions of the test are written (among others, the total amount of obstacles of each color), therefore, the counter update error also affects to the information displayed in the test conditions. Therefore, only the dodged red obstacles have been counted, so, a considerable difference is observed between the number of blue and red obstacles.

In conclusion, user results cannot be evaluated due to the number of factors that have negatively influenced the experience.

The main advantage of these tests has been the detection of unforeseen problems and obtaining opinions on the intuitiveness and ease to use. The first two games served as a contact for the user, so he still made mistakes due to lack of understanding about what he has to do. However, from the third game, the user understood the operation of the exercise and developed more naturally. The first two games served as a contact for the user, so he still made mistakes due to lack of understanding about what he has to do. However, from the third game, the user understood the operation of the exercise and developed more naturally.

Testing according to protocol

These tests include a 60-year-old woman with no virtual reality experience and a dominant right hand (player 2). Also, a 25-year-old man with no experience in virtual reality and a dominant left hand (player 3).

Player 2 has done all the tests while sitting down because of the nervousness he felt about trying virtual reality, it is possible that sitting up increases the difficulty of hitting the center of the obstacles and has been affected in the results.

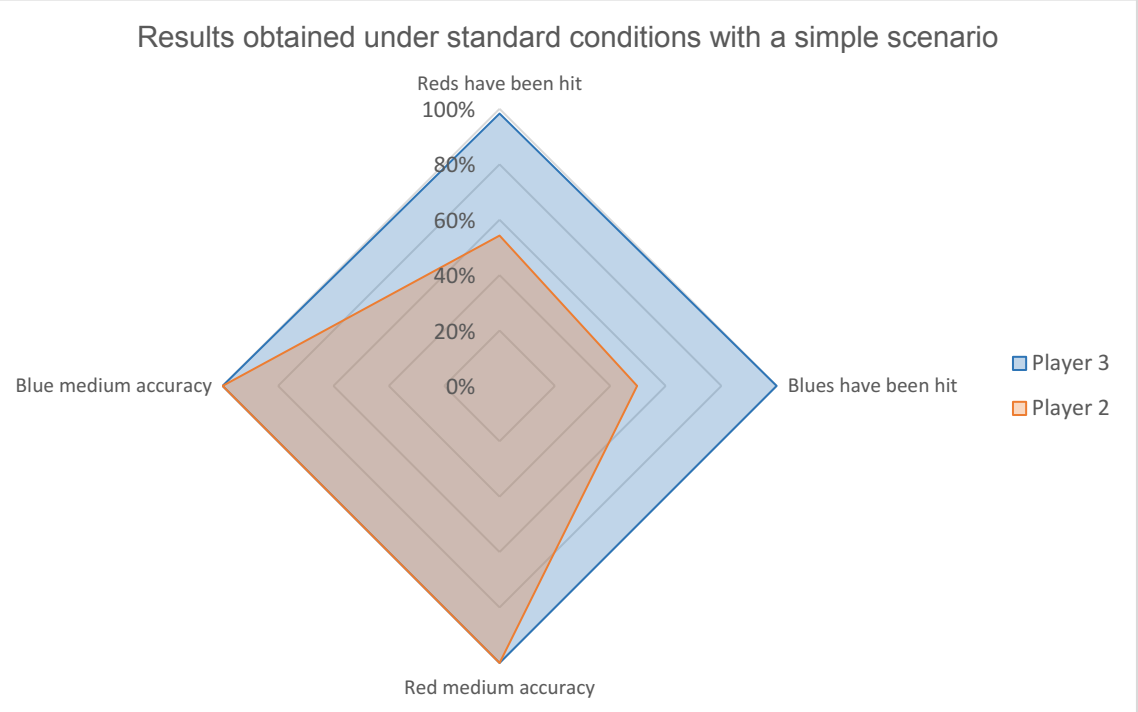
Evaluation under standard conditions

The results obtained with standard conditions will be used to compare the results of the players. It does not make sense to compare the results obtained with the other tests because each player uses different conditions.

Reminder: Blue obstacles are hit with the left hand and red obstacles with the right hand.

Simple scenario

Outcome variables	Player 2	Player 3
Blues have been hit	100,00%	100,00%
Blues have been hit incorrectly	7,14%	0,00%
Blues have been missed	0,00%	0,00%
Reds have been hit	100,00%	100,00%
Reds have been hit incorrectly	11,76%	0,00%
Reds have been missed	0,00%	0,00%
Totals have been hit	100,00%	100,00%
Totals have been missed	0,00%	0,00%
Blue medium accuracy	49,63%	100,00%
Red medium accuracy	54,26%	98,31%
Total average accuracy	52,17%	99,07%



Both players have hit all obstacles, this shows that the players maintain reflexes and ability to react to obstacles.

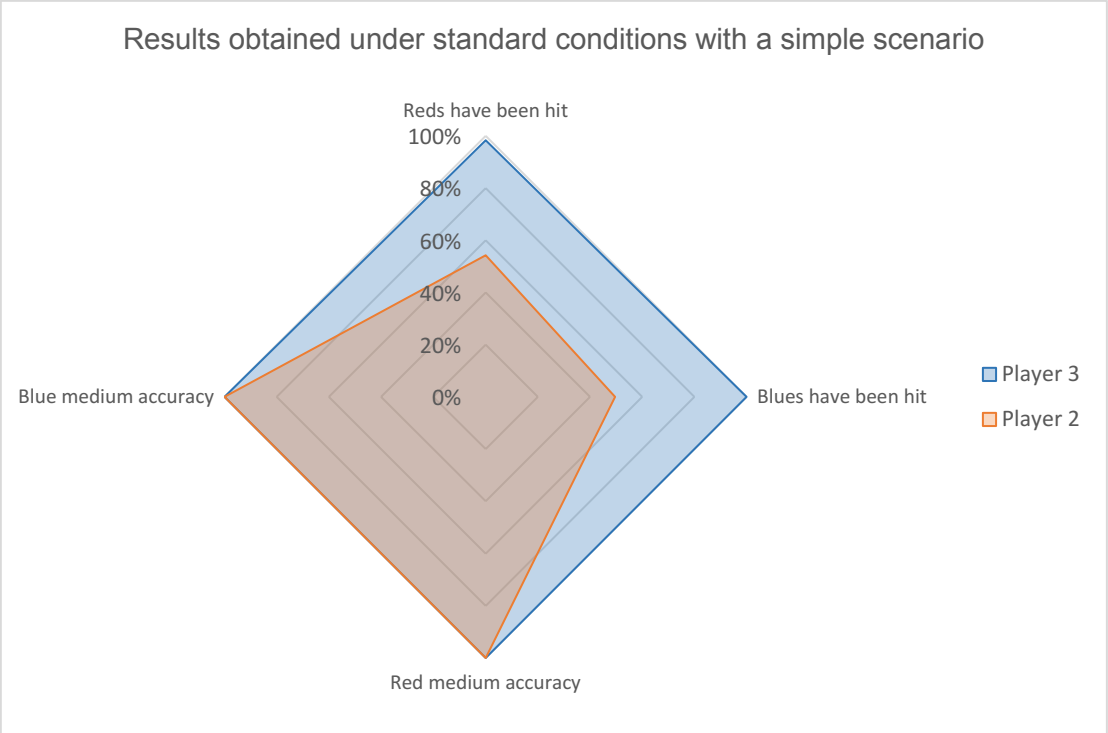
The player gets more precision in reds than blues, however the percentage of reds hit incorrectly is higher. This may be because the right-handed player has more accurately hit the red obstacles, but due to the lack of virtual reality experience he has incorrectly hit more reds.

Player 3 has obtained satisfactory results. He has not hit any obstacles incorrectly and there are also no obstacles that have escaped. In this scenario, the number of red obstacles is slightly higher than the blue ones, that is, there are more obstacles to hit with the right than with the left.

Graphically a considerable precision difference can be observed between the two players.

Complex scenario

Outcome variables	Player 2	Player 3
Blue has been hit correctly	100,00%	100,00%
Blues has been hit incorrectly	12,50%	0,00%
Blues target is missed	0,00%	0,00%
Reds has been hit correctly	100,00%	100,00%
Reds has been hit incorrectly	0,00%	0,00%
Reds target is missed	0,00%	0,00%
Total hit	100,00%	100,00%
Total not hit	0,00%	0,00%
Blue medium accuracy	76,24%	100,00%
Red medium accuracy	87,21%	100,00%
Total average accuracy	81,16%	100,00%



Player 2 hits 100% of the red obstacles but doesn´t hit 12.5% of the blue obstacles. In this scenario, you get a higher average precision than the simple one, it may be because the simple scenario has served as training for this second scenario.

Player 3 has a slightly higher number of blue obstacles than the red ones, that is, there are more obstacles to hit with the left than with the right. The average precision obtained in both hands in both scenarios exceeds 95%, therefore, it is considered that the player had no problems being left-handed and the interface seemed intuitive without any type of visual problem.

This time, Player 2 gets considerably better results than in the simple scenario. Graphically, less precision difference is observed than in the simple scenario.

Conclusions

The presented videogame to the players is intuitive and with pleasant controls. It has been seen that with this exercise you can obtain interesting information about the player's abilities in his current state with the disease.

This videogame is convenient to practice improving those reduced abilities of the player in a pleasant and entertaining way.

CONCLUSIONS

Introduction

The purpose of this section is showing the conclusions reached in the development of the project. The objectives achieved, the problems encountered, and the future line of the videogame will be explained.

Achievement of goals

The project started with the objective of developing a virtual reality prototype that simulates a rehabilitation exercise for patients with neurological disorders. Mainly, the videogame consists about the destruction of a sequence of obstacles that will be shown to the patient. The patient must touch each obstacle as quickly and accurately as possible.

Regarding the functional and non-functional requirements, it´s necessary to comply with all the requirements except for the sound, music and language requirements. This is because it´s considered that the player doesn't need to listen to music to perform the exercise.

For the creation of the videogame, the student has started with little experience in the programs used:

3DS MAX: It deals with the modeling of all the 3D objects that are shown on the stage of the videogame. The student has decided to design and model all three-dimensional objects although there are libraries and models of free and free use. The installation of plugins in the program has also been tested only in order to learn how to use them, however, no plugin has been used in its final models to get as much instruction as possible about this program. The objects modeled and textured with this program will be exported and later imported into the Unity program.

Unity: it deals with the creation of the videogame scene and its operation. This program has allowed the creation and import the objects inthe scene, insertion of background music, texturing of different elements, position of all three-dimensional objects, animation of objects and programming of operation.

The resulting videogame has been analyzed with an evaluation system that divides the system into independent criteria to increase the degree of evaluation precision. The videogame complies the desired objectives and has improved the desired result in some cases.

Also, the videogame has been tested with real people. The videogame has turned out to be intuitive and easy to use for tested people. From the tests carried out, it can be seen that this game has great potential to measure and obtain information about the physical state of the players.

At the level of personal growth, the student has learned to use a graphics engine. He has learned about the design and modeling of three-dimensional objects. All this based on total ignorance about the programs used and learning for hilmself.

Problems found

Different problems have been found:

- Videogame design: Initially, there was no clear and solid structure in mind about the form the videogame would take. Only the idea of destroying obstacles as close to the center as possible was clear, so it was difficult to establish the first design ideas. With the first design ideas could be made improvements
- Design of three-dimensional objects: creating a mental image of the object to be designed is a complicated task. The design of the objects has gone through different phases and prototypes to choose the shape and the possible deformations that it should have.
  All designs first took shape on paper to know what functionality to look for and learn in 3DS MAX. This methodology is very important because if you start without a mental image of the design and start looking for options directly in 3DS MAX, it can be stressful, since having so many options will make it difficult to choose which one to use for modeling.
- Unity Physics: Initially collider was not being used properly on obstacles. First, it was decided to create spherical collider that encompassed the obstacle itself. However, this caused the player to destroy the obstacle too soon and it was difficult to obtain good results. It was also tried to use two-dimensional “collider”, in this way, a circular “collider” could be created that included the obstacle and was destroyed when it really corresponds. However, two-dimensional "collider" has problems in Unity because internally non-functional in the same way as three-dimensional, then they never detected the collision and acted as indestructible obstacles to the user.
- Scene changes: When the player finished the game in any scenario (simple or complex) and automatically returned to the main menu, he appeared with 4 hands instead of 2. This problem occurred because the player instances never were eliminated from memory. Then, when changing scenes, new player instances were created and accumulated.
- Hit Score: Initially the player obtained a score per hit of the obstacle, based on the distance to the center (in two-dimensional coordinates) of the obstacle. The problem with this implementation is that the player would practically never get 100 points because it is very unlikely to hit exactly the point of the center two-dimensional coordinates. The score obtained was then on average even below 80% (in apparently good hitting cases). This is why a central rank was created where the player will always get 100 points if he hits within that range.
- 3DS MAX functionalities: Among the functionalities used in the 3DS MAX program, the most problematic have been folding an object, twisting an object and converting an object to "Editable Poly". The advantage of “Editable Poly” is that you can manipulate an object by sublevels or layers and you can also manipulate the positions of the segments of the object. The segments act as vertices of a polyhedron to give it the desired shape. However, learning how to use this was not easy, since it was not understood at first how to handle the layers that structured an object or how to add new layers on top of them.

Weak points

The videogame has some weaknesses:

- Obstacle Destruction: To destroy an obstacle the player must avoid contact of the upper part of the saber / wand with the obstacle. However, it has been seen the case of users who intuitively lower the saber / wand so that it disappears from their visual point and consider that this is how they are avoiding the obstacle.
- Immediately end a test: if the user gets tired of a test or has started a test with the wrong parameters, the system does not give the possibility to return to the main menu during the test, therefore, he has to wait for the test to finish .
- Player height: the developed videogame does not take into account the player's height. It would be advisable to create different versions of the videogame adjusting to the height of it.
- Updating of main menu parameters: when the user returns to the main menu after a game, the parameter values are again the standars. The user may intuitively think that the parameters of the last game played were displayed in the main menu.
- Stage Center: The complex stage uses a rocky path to indicate the route that obstacles will circulate in and the simple stage uses a white line. Paths may need to be modified to more accurately depict the route of obstacles. Also, the center of the main menu doesn't exactly match the center of the test scenarios, so they should also fit together.

Future lines

Currently the videogame is in a stable and functional version. If the weak points were fixed, it could be possible to use in health centers or hospitals to measure patient capacities.

At the future level, this videogame can be improved to measure more characteristics. It could enter a parameter that indicates the size of the obstacles or two parameters to establish a random range of obstacle size. It could also increase the number of possible obstacles spawn points. It could include “trap obstacles”, which the patient must dodge instead of hitting.

In short, this videogame can continue working to expand the results obtained by a patient.

Conclusions

Carrying out this project has been didactic for the student. The student has learned about the programs used for the development of the videogame and virtual reality.